Federated Learning (FL) Lab

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

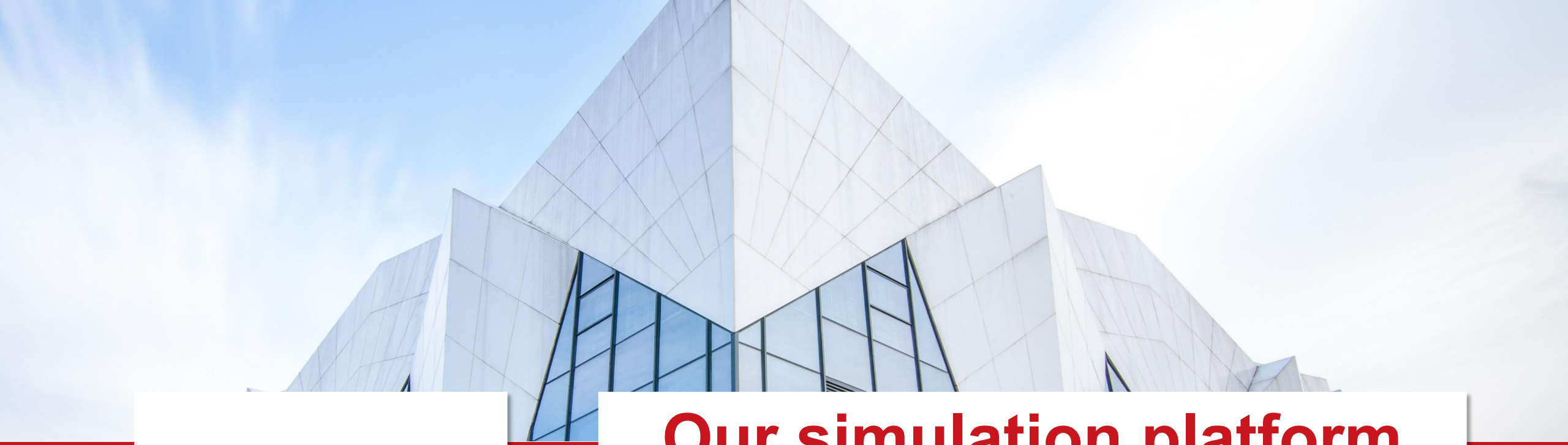
目录

1 Our simulation platform

2 Hands-on algorithms

3 Advanced algorithms

4 Other algorithms



01

Our simulation platform

- Overview
- Dataset generation
- System introduction
- FedAvg example

Our simulation platform

Overview

- Currently, 1.9k stars, 369 forks
- 39 traditional FL (tFL) and personalized FL (pFL) algorithms, 3 scenarios, and 24 datasets.
- Record the GPU memory usage for the model
- Differential privacy

Our simulation platform

Baselines

Traditional FL (tFL)

Basic tFL

- FedAvg — Communication-Efficient Learning of Deep Networks from Decentralized Data *AISTATS 2017*

Update-correction-based tFL

- SCAFFOLD - SCAFFOLD: Stochastic Controlled Averaging for Federated Learning *ICML 2020*

Regularization-based tFL

- FedProx — Federated Optimization in Heterogeneous Networks *MLsys 2020*
- FedDyn — Federated Learning Based on Dynamic Regularization *ICLR 2021*

Model-splitting-based tFL

- MOON — Model-Contrastive Federated Learning *CVPR 2021*
- FedLC — Federated Learning With Label Distribution Skew via Logits Calibration *ICML 2022*

Knowledge-distillation-based tFL

- FedGen — Data-Free Knowledge Distillation for Heterogeneous Federated Learning *ICML 2021*
- FedNTD — Preservation of the Global Knowledge by Not-True Distillation in Federated Learning *NeurIPS 2022*

Heuristically-search-based tFL

- FedCross - FedCross: Towards Accurate Federated Learning via Multi-Model Cross-Aggregation *ICDE 2024*

Our simulation platform

Baselines

> Personalized FL (pFL)

Meta-learning-based pFL

- Per-FedAvg — Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach *NeurIPS 2020*

Regularization-based pFL

- pFedMe — Personalized Federated Learning with Moreau Envelopes *NeurIPS 2020*
- Ditto — Ditto: Fair and robust federated learning through personalization *ICML 2021*

Personalized-aggregation-based pFL

- APFL — Adaptive Personalized Federated Learning *2020*
- FedFomo — Personalized Federated Learning with First Order Model Optimization *ICLR 2021*
- FedAMP — Personalized Cross-Silo Federated Learning on non-IID Data *AAAI 2021*
- FedPHP — FedPHP: Federated Personalization with Inherited Private Models *ECML PKDD 2021*
- APPLE — Adapt to Adaptation: Learning Personalization for Cross-Silo Federated Learning *IJCAI 2022*
- FedALA — FedALA: Adaptive Local Aggregation for Personalized Federated Learning *AAAI 2023*

Model-splitting-based pFL

- FedPer — Federated Learning with Personalization Layers *2019*
- LG-FedAvg — Think Locally, Act Globally: Federated Learning with Local and Global Representations *2020*
- FedRep — Exploiting Shared Representations for Personalized Federated Learning *ICML 2021*
- FedRoD — On Bridging Generic and Personalized Federated Learning for Image Classification *ICLR 2022*

Our simulation platform

Datasets

Datasets and scenarios (updating)

We support 3 types of scenarios with various datasets and move the common dataset splitting code into `./dataset/utils` for easy extension. If you need another data set, just write another code to download it and then use the utils.

label skew scenario

For the ***label skew*** scenario, we introduce **16** famous datasets:

- MNIST
- EMNIST
- FEMNIST
- Fashion-MNIST
- Cifar10
- Cifar100
- AG News
- Sogou News
- Tiny-ImageNet
- Country211
- Flowers102
- GTSRB
- Shakespeare
- Stanford Cars
- COVIDx
- kvasir

Our simulation platform

Datasets

The datasets can be easily split into **IID** and **non-IID** versions. In the **non-IID** scenario, we distinguish between two types of distribution:

1. **Pathological non-IID**: In this case, each client only holds a subset of the labels, for example, just 2 out of 10 labels from the MNIST dataset, even though the overall dataset contains all 10 labels. This leads to a highly skewed distribution of data across clients.
2. **Practical non-IID**: Here, we model the data distribution using a Dirichlet distribution, which results in a more realistic and less extreme imbalance. For more details on this, refer to this paper.

Additionally, we offer a `balance` option, where data amount is evenly distributed across all clients.

feature shift scenario

For the ***feature shift*** scenario, we utilize **3** widely used datasets in Domain Adaptation:

- **Amazon Review** (raw data can be fetched from this link)
- **Digit5** (raw data available here)
- **DomainNet**

real-world scenario

For the ***real-world*** scenario, we introduce **5** naturally separated datasets:

- **Camelyon17** (5 hospitals, 2 labels)
- **iWildCam** (194 camera traps, 158 labels)
- **Omniglot** (20 clients, 50 labels)
- **HAR (Human Activity Recognition)** (30 clients, 6 labels)
- **PAMAP2** (9 clients, 12 labels)

Our simulation platform

Models

Models

- for MNIST and Fashion-MNIST
  i. Mclr_Logistic(1*28*28) # convex
  ii. LeNet()
  iii. DNN(1*28*28, 100)
- for Cifar10, Cifar100 and Tiny-ImageNet
  i. Mclr_Logistic(3*32*32) # convex
  ii. FedAvgCNN()
  iii. DNN(3*32*32, 100)
  iv. ResNet18, AlexNet, MobileNet, GoogleNet, etc.
- for AG_News and Sogou_News
  - LSTM()
  - fastText() in Bag of Tricks for Efficient Text Classification
  - TextCNN() in Convolutional Neural Networks for Sentence Classification
  - TransformerModel() in Attention is all you need
- for AmazonReview
  - AmazonMLP() in Curriculum manager for source selection in multi-source domain adaptation
- for Omniglot
  - FedAvgCNN()
- for HAR and PAMAP
  - HARCNN() in Convolutional neural networks for human activity recognition using mobile sensors

Our simulation platform

Installation

Environments

Install CUDA.

Install conda latest and activate conda.

For additional configurations, refer to the `prepare.sh` script.

```
conda env create -f env_cuda_latest.yaml  # Downgrade torch via pip if needed to match the CUDA version
```

Our simulation platform

Dataset

Examples for MNIST in the *label skew* scenario

```
cd ./dataset
# Please modify train_ratio and alpha in dataset\utils\dataset_utils.py

python generate_MNIST.py iid - - # for iid and unbalanced scenario
python generate_MNIST.py iid balance - # for iid and balanced scenario
python generate_MNIST.py noniid - pat # for pathological noniid and unbalanced scenario
python generate_MNIST.py noniid - dir # for practical noniid and unbalanced scenario
python generate_MNIST.py noniid - exdir # for Extended Dirichlet strategy
```

The command line output of running `python generate_MNIST.py noniid - dir`

```
Number of classes: 10
Client 0        Size of data: 2630      Labels:  [0 1 4 5 7 8 9]
                Samples of labels:  [(0, 140), (1, 890), (4, 1), (5, 319), (7, 29), (8, 1067), (9, 184)]
--------------------------------------------------
Client 1        Size of data: 499       Labels:  [0 2 5 6 8 9]
                Samples of labels:  [(0, 5), (2, 27), (5, 19), (6, 335), (8, 6), (9, 107)]
--------------------------------------------------
Client 2        Size of data: 1630      Labels:  [0 3 6 9]
                Samples of labels:  [(0, 3), (3, 143), (6, 1461), (9, 23)]
--------------------------------------------------
```

Our simulation platform

Execution

How to start simulating (examples for FedAvg)

- Download this project to an appropriate location using git.

```
git clone https://github.com/TsingZ0/PFLlib.git
```

- Create proper environments (see Environments).
- Build evaluation scenarios (see Datasets and scenarios (updating)).
- Run evaluation:

```
cd ./system
python main.py -data MNIST -m CNN -algo FedAvg -gr 2000 -did 0 # using the MNIST dataset, the FedAvg
python main.py -data MNIST -m CNN -algo FedAvg -gr 2000 -did 0,1,2,3 # running on multiple GPUs
```

Note: It is preferable to tune algorithm-specific hyper-parameters before using any algorithm on a new machine.

Our simulation platform

- Top-down
  - Server: Aggregate, Send, Receive, Evaluate
    - Add
  - Client: Train, Receive, Send
    - Dataset
    - Model
    - Optimizer

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

02

Hands-on algorithms

- MOON
- FedDyn
- KT-pFL
- FedMA

MOON

MOON algorithm

Observation

- The global model trained on a whole dataset can learn a better representation than the local model trained on a skewed subset.
- Propose model-contrastive learning (MOON), which corrects the local updates by maximizing the agreement of representation learned by the current local model and the representation learned by the global model.

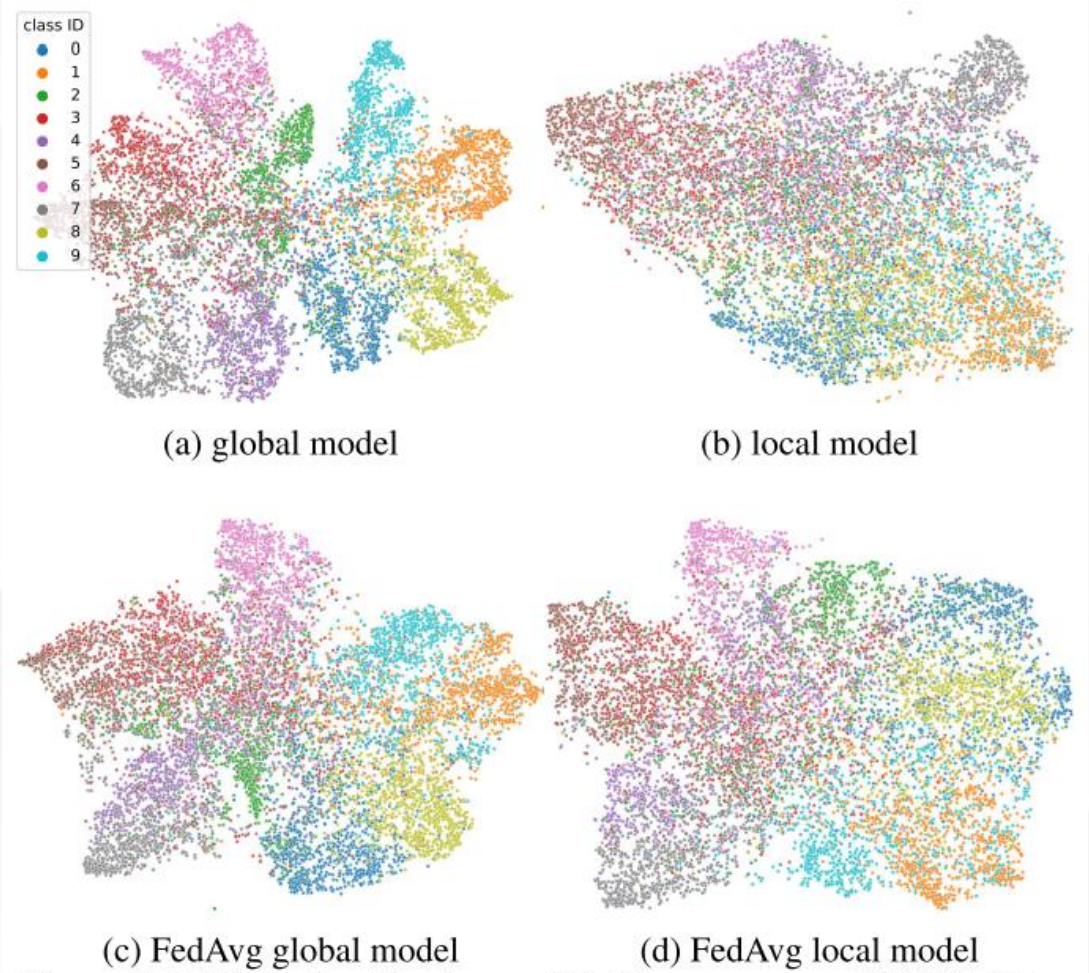


Figure 2. T-SNE visualizations of hidden vectors on CIFAR-10.

MOON algorithm

- Contrastive learning
  - The key idea of contrastive learning is to reduce the distance between the representations of different augmented views of the same image (i.e., positive pairs), and increase the distance between the representations of augmented views of different images (i.e., negative pairs)

$$l_{i,j} = -\log \frac{\exp(\text{sim}(x_i, x_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} \exp(\text{sim}(x_i, x_k)/\tau)}$$

MOON algorithm

- MOON
  - Global representation $z_{glob} = R_{w^t}(x)$
  - Local representation w_i^{t-1} (i.e., $z_{prev} = R_{w_i^{t-1}}(x)$)
  - Current representation $z = R_{w_i^t}(x)$
  -

$$\ell_{con} = -\log \frac{\exp(\text{sim}(z, z_{glob})/\tau)}{\exp(\text{sim}(z, z_{glob})/\tau) + \exp(\text{sim}(z, z_{prev})/\tau)}$$

$$\ell = \ell_{sup}(w_i^t; (x, y)) + \mu \ell_{con}(w_i^t; w_i^{t-1}; w^t; x),$$

$$\min_{w_i^t} \mathbb{E}_{(x,y)\sim D^i} [\ell_{sup}(w_i^t; (x, y)) + \mu \ell_{con}(w_i^t; w_i^{t-1}; w^t; x)].$$

MOON algorithm

MOON

Algorithm 1: The MOON framework

Input: number of communication rounds T, number of parties N, number of local epochs E, temperature τ, learning rate η, hyper-parameter μ

Output: The final model w^T

Server executes:
initialize w^0
for $t = 0, 1, ..., T-1$ **do**
  for $i = 1, 2, ..., N$ **in parallel do**
    send the global model w^t to P_i
    $w_i^t \leftarrow$ **PartyLocalTraining**(i, w^t)
  $w^{t+1} \leftarrow \sum_{k=1}^{N} \frac{|\mathcal{D}^i|}{|\mathcal{D}|} w_k^t$
return w^T

PartyLocalTraining(i, w^t):
$w_i^t \leftarrow w^t$
for epoch $i = 1, 2, ..., E$ **do**
  for each batch $\mathbf{b} = \{x, y\}$ of $\mathcal{D}^i$ **do**
    $\ell_{sup} \leftarrow CrossEntropyLoss(F_{w_i^t}(x), y)$
    $z \leftarrow R_{w_i^t}(x)$
    $z_{glob} \leftarrow R_{w^t}(x)$
    $z_{prev} \leftarrow R_{w_i^{t-1}}(x)$
    $\ell_{con} \leftarrow -\log \frac{\exp(\text{sim}(z, z_{glob})/\tau)}{\exp(\text{sim}(z, z_{glob})/\tau) + \exp(\text{sim}(z, z_{prev})/\tau)}$
    $\ell \leftarrow \ell_{sup} + \mu \ell_{con}$
    $w_i^t \leftarrow w_i^t - \eta \nabla \ell$
return w_i^t to server

FedDyn

FedDyn algorithm

- Intuition
  - Training models on local data that minimize local empirical loss appears to be meaningful, but yet, doing so is fundamentally inconsistent with minimizing the global empirical loss.
  - Dynamically modify the device objective with a penalty term so that, in the limit, when model parameters converge, they do so to stationary points of the global empirical loss.

FedDyn algorithm

FedDyn

Algorithm 1: Federated Dynamic Regularizer - (FedDyn)

Input: $T, \boldsymbol{\theta}^0, \alpha > 0, \nabla L_k(\boldsymbol{\theta}_k^0) = \mathbf{0}$.

for $t = 1, 2, \ldots T$ **do**

  Sample devices $\mathcal{P}_t \subseteq [m]$ and transmit $\boldsymbol{\theta}^{t-1}$ to each selected device,

  for *each device* $k \in \mathcal{P}_t$, *and in parallel* **do**

    Set $\boldsymbol{\theta}_k^t = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} L_k(\boldsymbol{\theta}) - \langle \nabla L_k(\boldsymbol{\theta}_k^{t-1}), \boldsymbol{\theta} \rangle + \frac{\alpha}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{t-1}\|^2$,

    Set $\nabla L_k(\boldsymbol{\theta}_k^t) = \nabla L_k(\boldsymbol{\theta}_k^{t-1}) - \alpha \left(\boldsymbol{\theta}_k^t - \boldsymbol{\theta}^{t-1}\right)$,

    Transmit device model $\boldsymbol{\theta}_k^t$ to server,

  end for

  for *each device* $k \notin \mathcal{P}_t$, *and in parallel* **do**

    Set $\boldsymbol{\theta}_k^t = \boldsymbol{\theta}_k^{t-1}, \nabla L_k(\boldsymbol{\theta}_k^t) = \nabla L_k(\boldsymbol{\theta}_k^{t-1})$,

  end for

  Set $\boldsymbol{h}^t = \boldsymbol{h}^{t-1} - \alpha \frac{1}{m} \left(\sum_{k \in \mathcal{P}_t} \boldsymbol{\theta}_k^t - \boldsymbol{\theta}^{t-1}\right)$,

  Set $\boldsymbol{\theta}^t = \left(\frac{1}{|\mathcal{P}_t|} \sum_{k \in \mathcal{P}_t} \boldsymbol{\theta}_k^t\right) - \frac{1}{\alpha} \boldsymbol{h}^t$

end for

FedDyn algorithm

- Analysis

$$\boldsymbol{\theta}_k^t = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\mathfrak{R}_k(\boldsymbol{\theta}; \boldsymbol{\theta}_k^{t-1}, \boldsymbol{\theta}^{t-1}) \triangleq L_k(\boldsymbol{\theta}) - \langle \nabla L_k(\boldsymbol{\theta}_k^{t-1}), \boldsymbol{\theta} \rangle + \frac{\alpha}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}^{t-1}\|^2 \right]$$

- The first order condition

$$\nabla L_k(\boldsymbol{\theta}_k^t) - \nabla L_k(\boldsymbol{\theta}_k^{t-1}) + \alpha(\boldsymbol{\theta}_k^t - \boldsymbol{\theta}^{t-1}) = \mathbf{0}$$

- If local device models converge, they converge to the server model, and the convergence point is a stationary point of the global loss.

if $\boldsymbol{\theta}_k^t \to \boldsymbol{\theta}_k^\infty$, it generally follows that, $\nabla L_k(\boldsymbol{\theta}_k^t) \to \nabla L_k(\boldsymbol{\theta}_k^\infty)$, and as a consequence, we have $\boldsymbol{\theta}^t \to \boldsymbol{\theta}_k^\infty$. In turn this implies that $\boldsymbol{\theta}_k^\infty \to \boldsymbol{\theta}^\infty$, i.e., is independent of k.

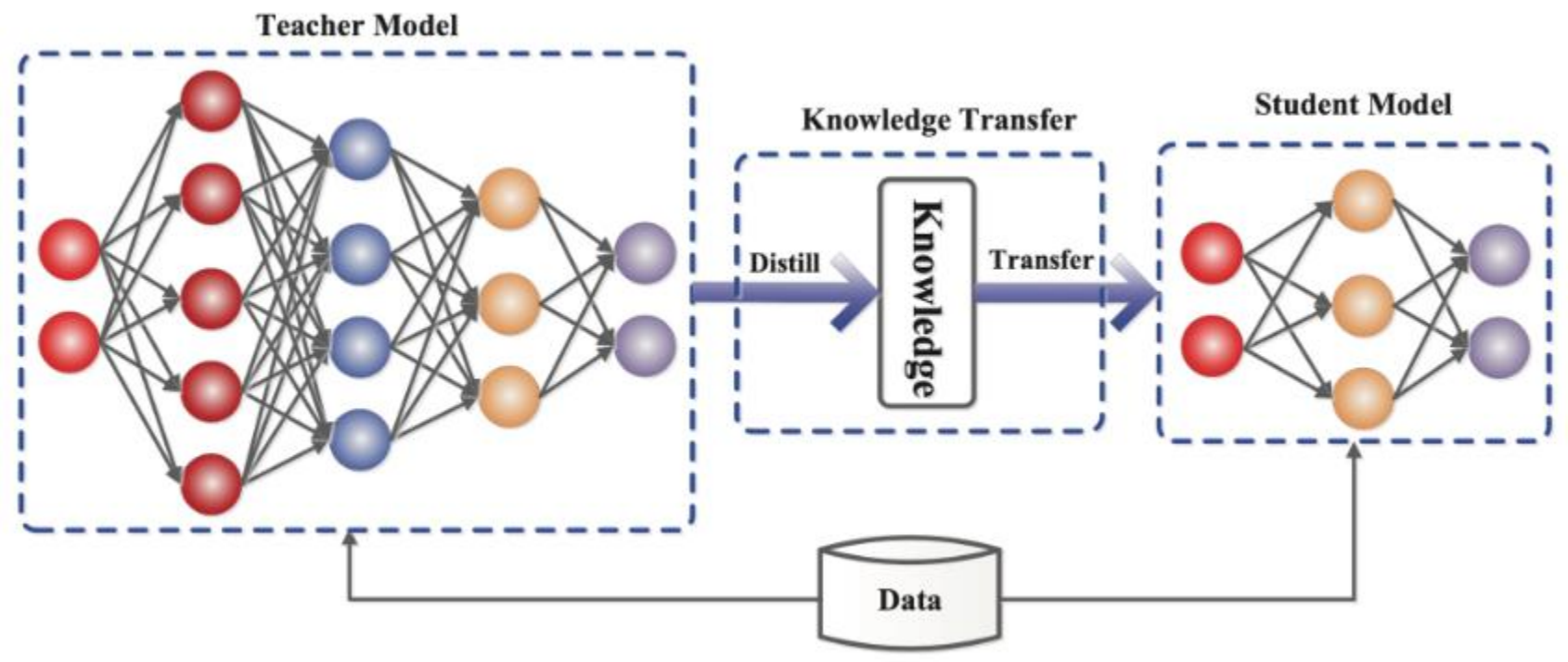
KT-pFL

KT-pFL algorithm

- Intuition
  - Main idea is to allow each client to maintain a personalized soft prediction at the server that can be updated by a linear combination of all clients local soft predictions using a knowledge coefficient matrix.
  - Regardless of model structures

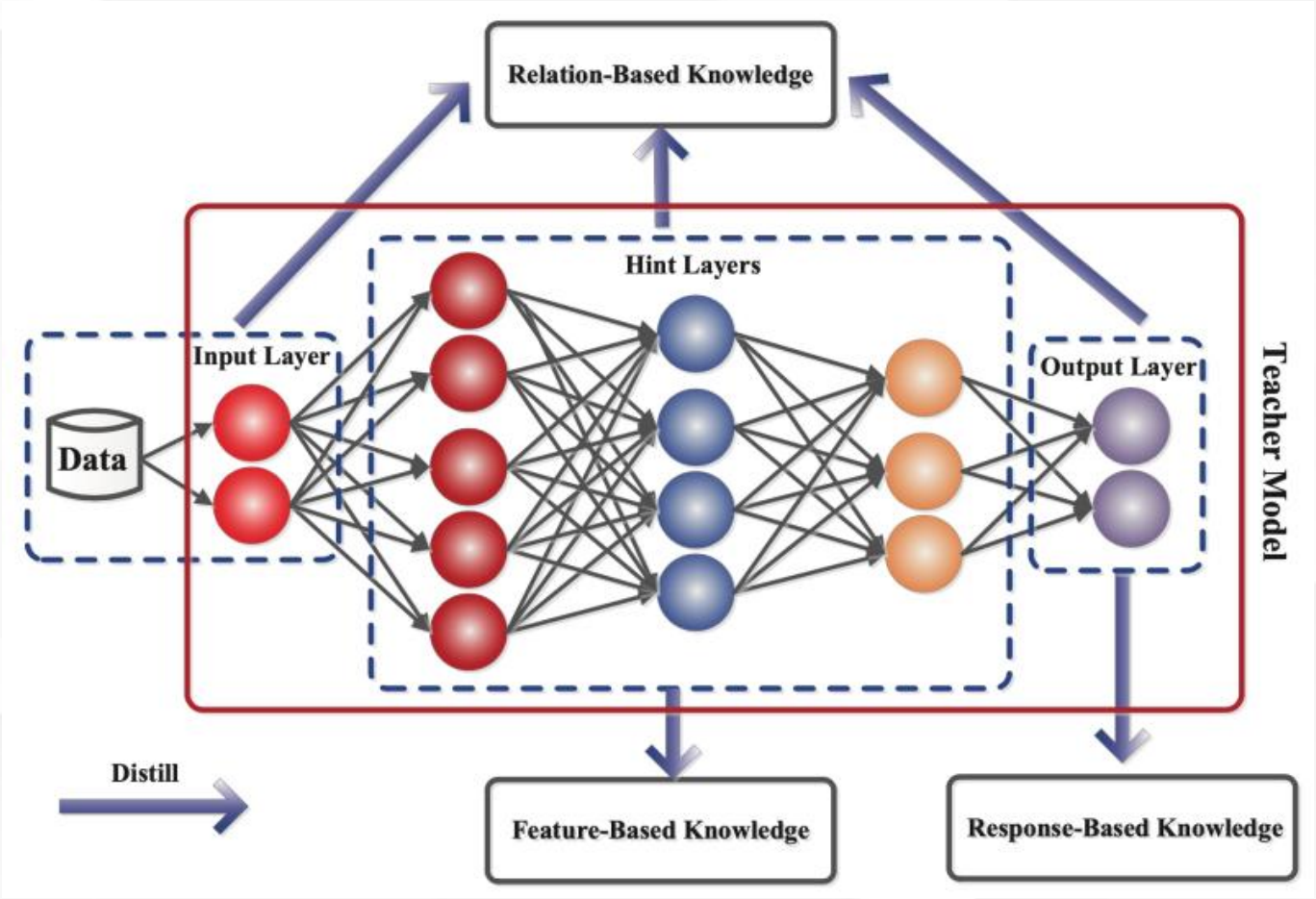
KT-pFL algorithm

- Knowledge Distillation (KD)
  - Transfer knowledge from well-learned teacher model to student model



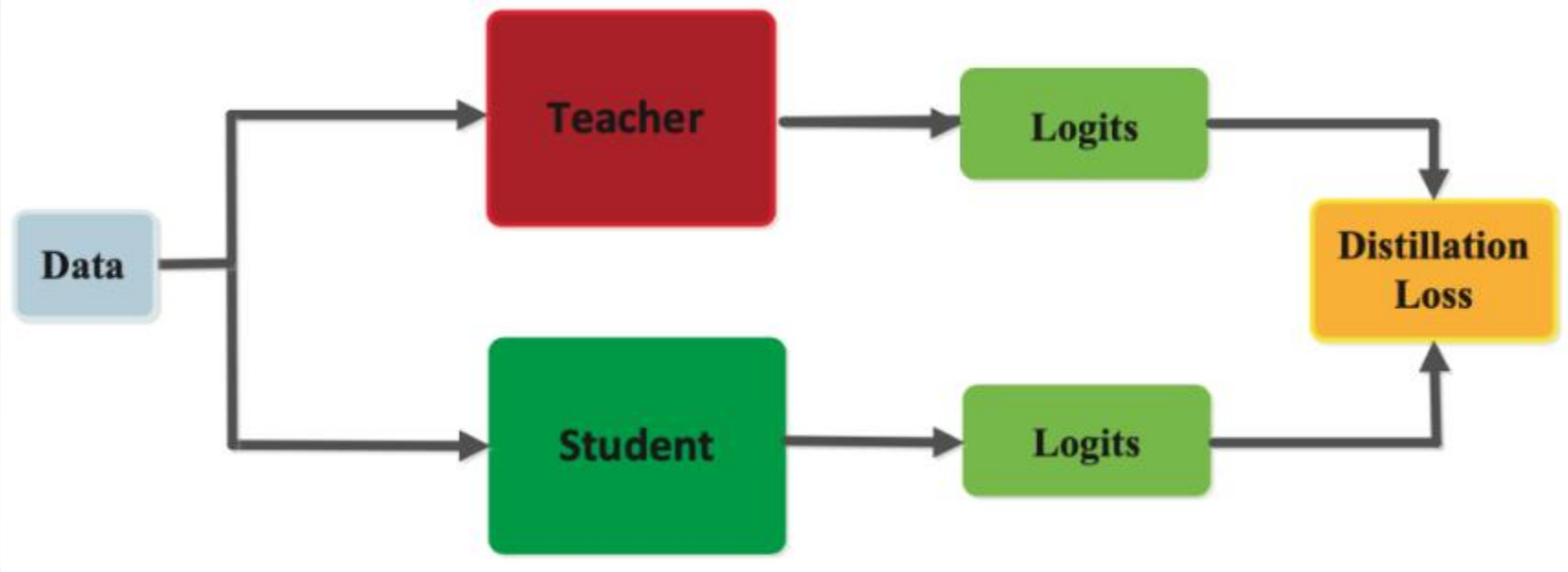
KT-pFL algorithm

- Knowledge Distillation (KD)
  - Classification



KT-pFL algorithm

- Knowledge Distillation (KD)
  - Response-based KD



KT-pFL algorithm

- KT-pFL
  - Objective

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_n(\mathbf{w}), \text{ where } \mathcal{L}_n(\mathbf{w}) = \frac{1}{D_n} \sum_{i=1}^{D_n} \mathcal{L}_{CE}(\mathbf{w}; x_i, y_i)$$

$$\min_{\mathbf{w}^1,\cdots,\mathbf{w}^N} \mathcal{L}(\mathbf{w}^1, \cdots, \mathbf{w}^N) := \sum_{n=1}^{N} \frac{D_n}{D} L_n(\mathbf{w}^n)$$

KT-pFL algorithm

- KT-pFL
  - Personalized loss function
    - Kullback–Leibler (KL) Divergence
    - c_{mn} is the knowledge coefficient which is used to estimate the contribution from client m to n.
    - $s(w^n, \hat{x})$ can be deemed to be a soft prediction of the client n

$$\mathcal{L}_{per,n}(\mathbf{w}^n) := \mathcal{L}_n(\mathbf{w}^n) + \lambda \sum_{\hat{x} \in \mathbb{D}_r} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} c_{mn} \cdot s(\mathbf{w}^m, \hat{x}), s(\mathbf{w}^n, \hat{x})\right)$$

$$s(\mathbf{w}^n, \hat{x}) = \frac{\exp(z_c^n/T)}{\sum_{c=1}^{C} \exp(z_c^n/T)},$$

KT-pFL algorithm

- KT-pFL
  - Knowledge coefficient matrix

$$\mathbf{c} = \begin{Bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{Bmatrix}.$$

KT-pFL algorithm

- KT-pFL
  - Objective

$$\min_{\mathbf{w},\mathbf{c}} \mathcal{L}(\mathbf{w},\mathbf{c}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_{per,n}(\mathbf{w}^n) + \rho \|\mathbf{c} - \frac{\mathbf{1}}{N}\|^2,$$

$$\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N] \in \mathbb{R}^{\sum_{n=1}^{N} d_n}$$

KT-pFL algorithm

- KT-pFL
  - Training
    - Update w
      - Local Training
      - Distillation
    - Update c

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_1 \nabla_{\mathbf{w}^n} \mathcal{L}_n(\mathbf{w}^n; \xi_n),$$

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_2 \nabla_{\mathbf{w}^n} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} \mathbf{c}_m^{*,T} \cdot s(\mathbf{w}^m, \xi_r), s(\mathbf{w}^n, \xi_r)\right)$$

$$\mathbf{c} \leftarrow \mathbf{c} - \eta_3 \lambda \sum_{n=1}^{N} \frac{D_n}{D} \nabla_{\mathbf{c}} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} \mathbf{c}_m \cdot s(\mathbf{w}^{m,*}, \xi_r), s(\mathbf{w}^{n,*}, \xi_r)\right) - 2\eta_3 \rho(\mathbf{c} - \frac{\mathbf{1}}{N})$$

KT-pFL algorithm

- KT-pFL

Algorithm 1 KT-pFL Algorithm

Input: $\mathbb{D}, \mathbb{D}_r, \eta_1, \eta_2, \eta_3$ and T
Output: $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N]$

Initialize $\mathbf{w}_0$ and $\mathbf{c}_0$
procedure SERVER-SIDE OPTIMIZATION
    Distribute $\mathbf{w}_0$ and $\mathbf{c}_0$ to each client
    for each communication round $t \in \{1, 2, ..., T\}$ **do**
        for each client n **in parallel do**
            $\mathbf{w}_{t+1}^n \leftarrow ClientLocalUpdate(n, \mathbf{w}_t^n, \mathbf{c}_{t,n})$
        Update knowledge coefficient matrix $\mathbf{c}$ via (7)
        Distribute $\mathbf{c}_{t+1}$ to all clients
procedure CLIENTLOCALUPDATE$(n, \mathbf{w}_t^n, \mathbf{c}_{t,n})$
    Client n receives $\mathbf{w}_t^n$ and $\mathbf{c}_n$ from the server
    for each local epoch i from 1 to E **do**
        for mini-batch $\xi_t \subseteq \mathbb{D}_n$ **do**
            Local Training: update model parameters on private data via (5)
    for each distillation step j from 1 to R **do**
        for mini-batch $\xi_{r,t} \subseteq \mathbb{D}_r$ **do**
            Distillation: update model parameters on public data via (6)
    return local parameters $\mathbf{w}_{t+1}^n$

KT-pFL algorithm

KT-pFL

- Illustration

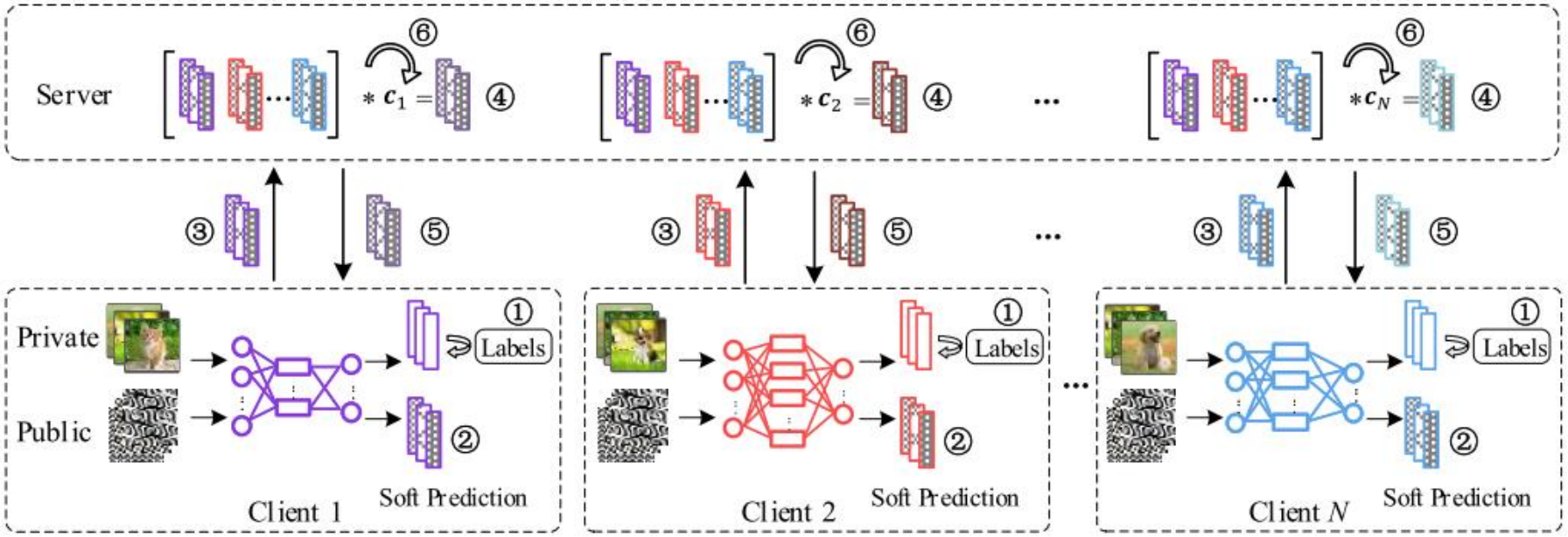
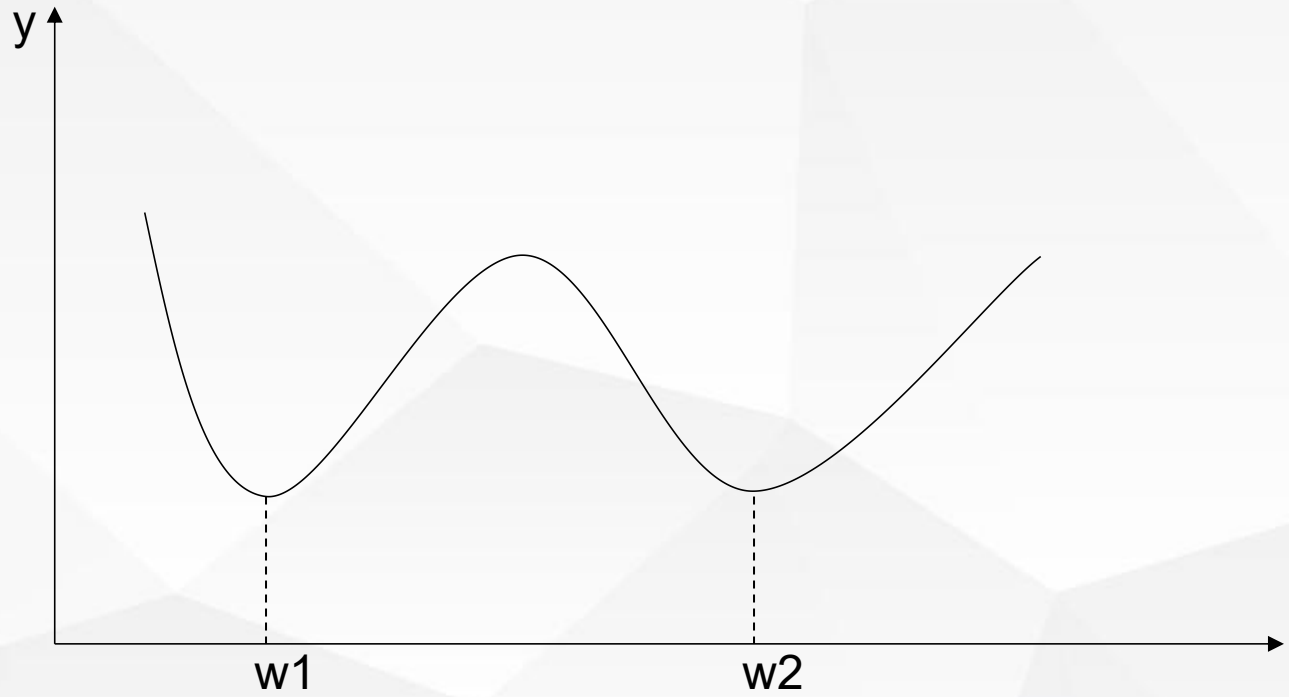


Figure 1: Illustration of the KT-pFL framework. The workflow includes 6 steps: ① local training on private data; ②, ③ each client outputs the local soft prediction on public data and sends it to the server; ④ the server calculates each client's personalized soft prediction via a linear combination of local soft predictions and knowledge coefficient matrix; ⑤ each client downloads the personalized soft prediction to perform distillation phase; ⑥ the server updates the knowledge coefficient matrix.

FedMA

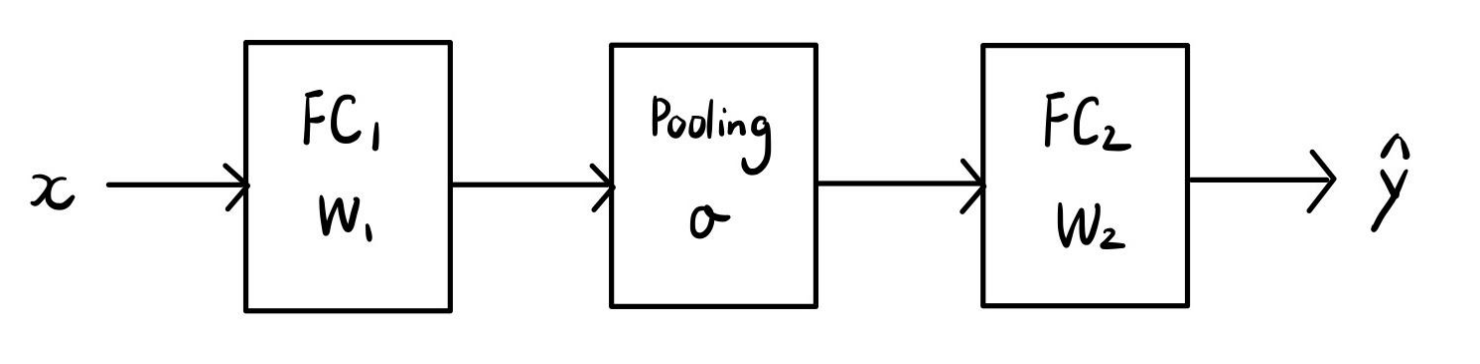
FedMA algorithm

- Permutation invariance



FedMA algorithm

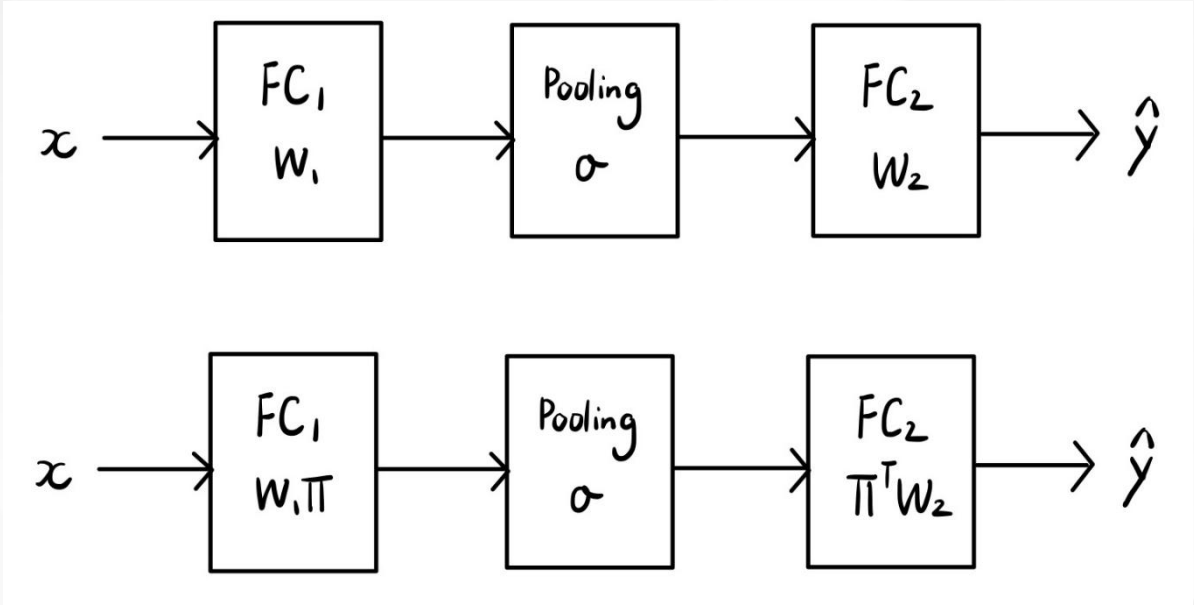
- Permutation invariance (fully-connected (FC) layer)



$$\hat{y} = \sigma(xW_1)W_2$$

FedMA algorithm

- Permutation invariance (fully-connected (FC) layer)



$$\hat{y} = \sigma(xW_1\Pi)\Pi^T W_2, \text{ where } \Pi \text{ is any } L \times L \text{ permutation matrix.}$$

FedMA algorithm

- Permutation invariance (fully-connected (FC) layer)

Client A:

$$\{W_1\Pi_j, \Pi_j^T W_2\}$$

Client B:

$$\{W_1\Pi_{j'}, \Pi_{j'}^T W_2\}$$

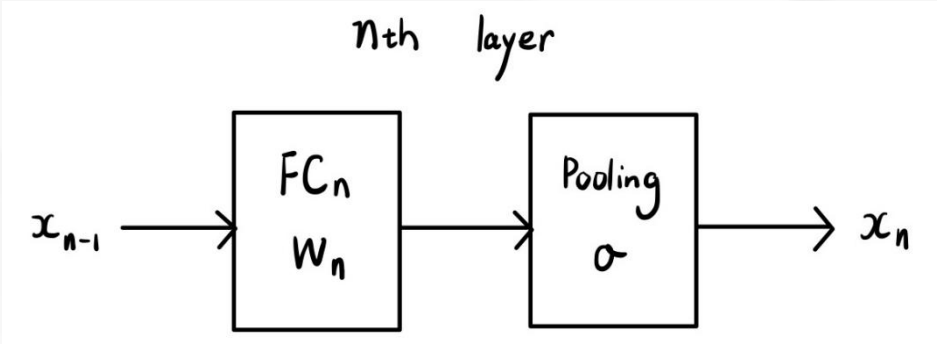
Aggregation

$$(W_1\Pi_j + W_1\Pi_{j'})/2 \neq W_1\Pi \text{ for any } \Pi$$

Solution: $(W_1\Pi_j\Pi_j^T + W_1\Pi_{j'}\Pi_{j'}^T)/2 = W_1$

FedMA algorithm

- Permutation invariance (FCs)

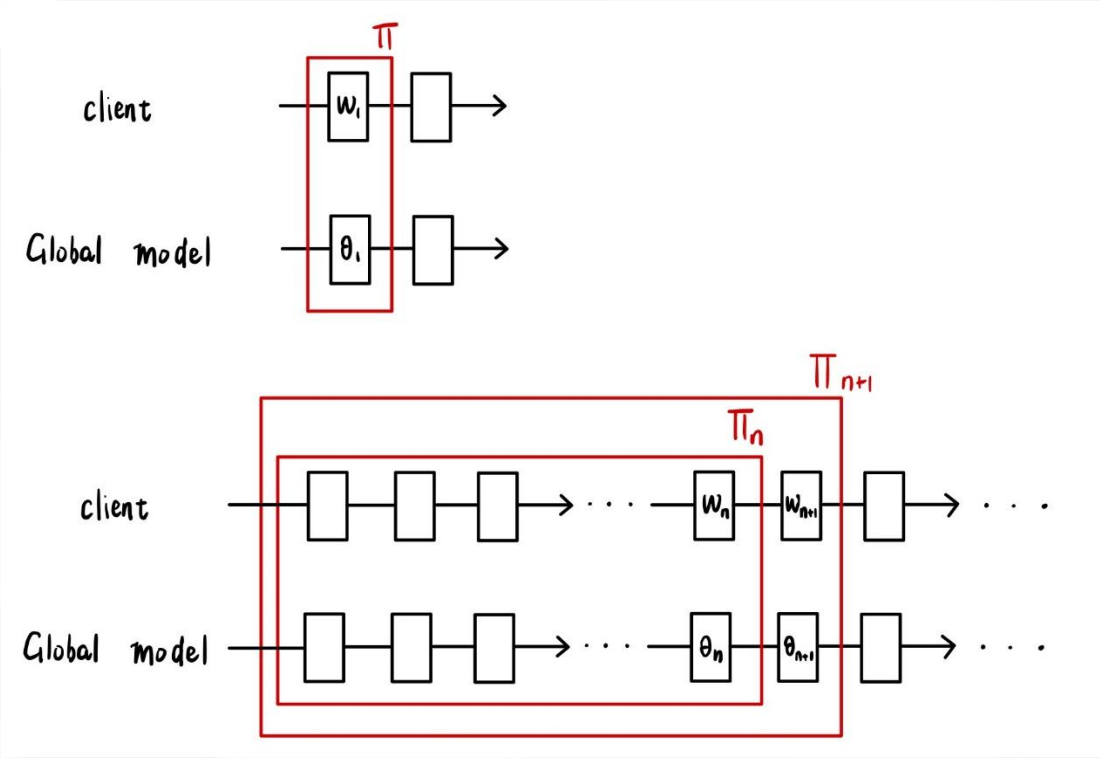


Simple FCs: $\hat{y} = \sigma(xW_1\Pi)\Pi^T W_2$

Deep FCs: $x_n = \sigma(x_{n-1}\Pi_{n-1}^T W_n \Pi_n)$

FedMA algorithm

- Permutation invariance (FCs)



$$x_n = \sigma(x_{n-1}\Pi_{n-1}^T W_n \Pi_n)$$

FedMA algorithm

- Permutation invariance （CNN)

FC : $x_n = \sigma(x_{n-1}\Pi_{n-1}^T W_n \Pi_n)$

CNN : $x_n = \sigma(\mathbf{Conv}(x_{n-1}, \Pi_{n-1}^T W_n \Pi_n))$

FedMA algorithm

- Permutation invariance (recall)

Client A:

$$\{W_1\Pi_j, \Pi_j^T W_2\}$$

Client B:

$$\{W_1\Pi_{j'}, \Pi_{j'}^T W_2\}$$

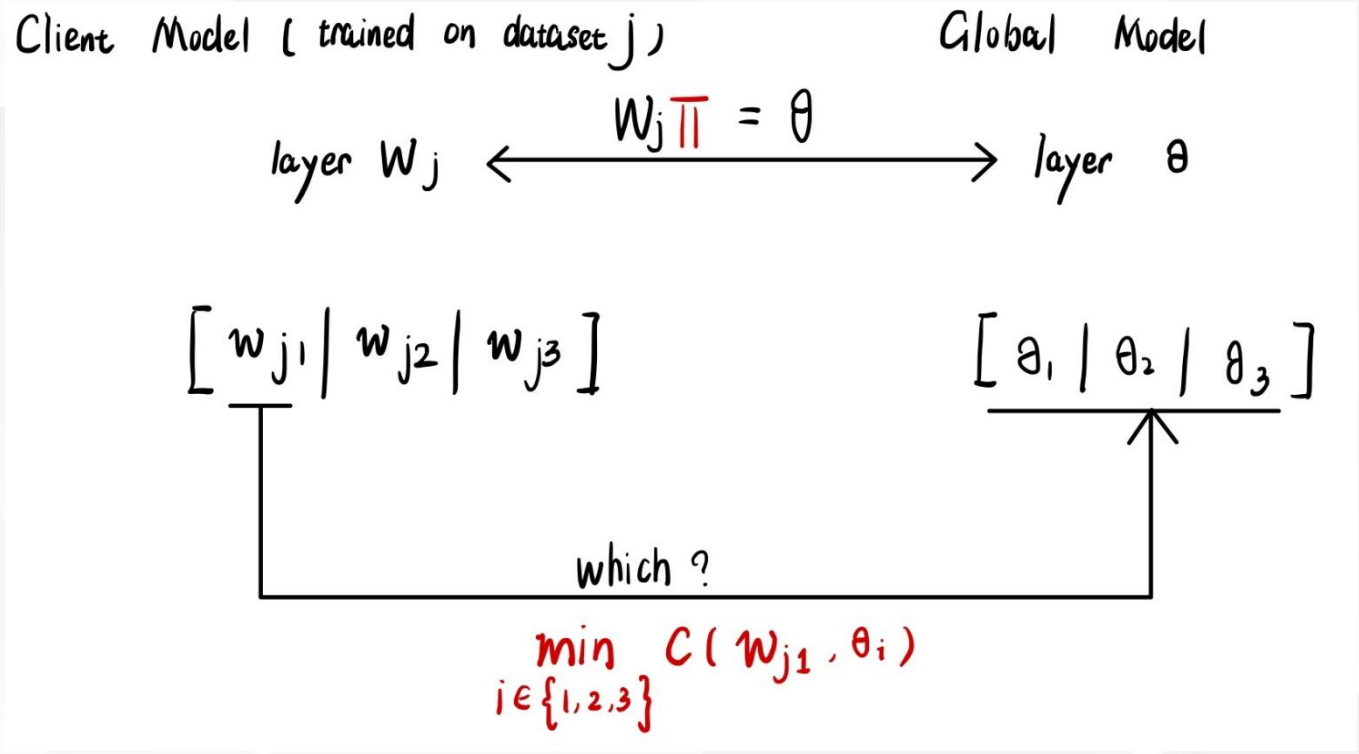
Aggregation

$$(W_1\Pi_j + W_1\Pi_{j'})/2 \neq W_1\Pi \text{ for any } \Pi$$

Solution: $(W_1\Pi_j\Pi_j^T + W_1\Pi_{j'}\Pi_{j'}^T)/2 = W_1$

FedMA algorithm

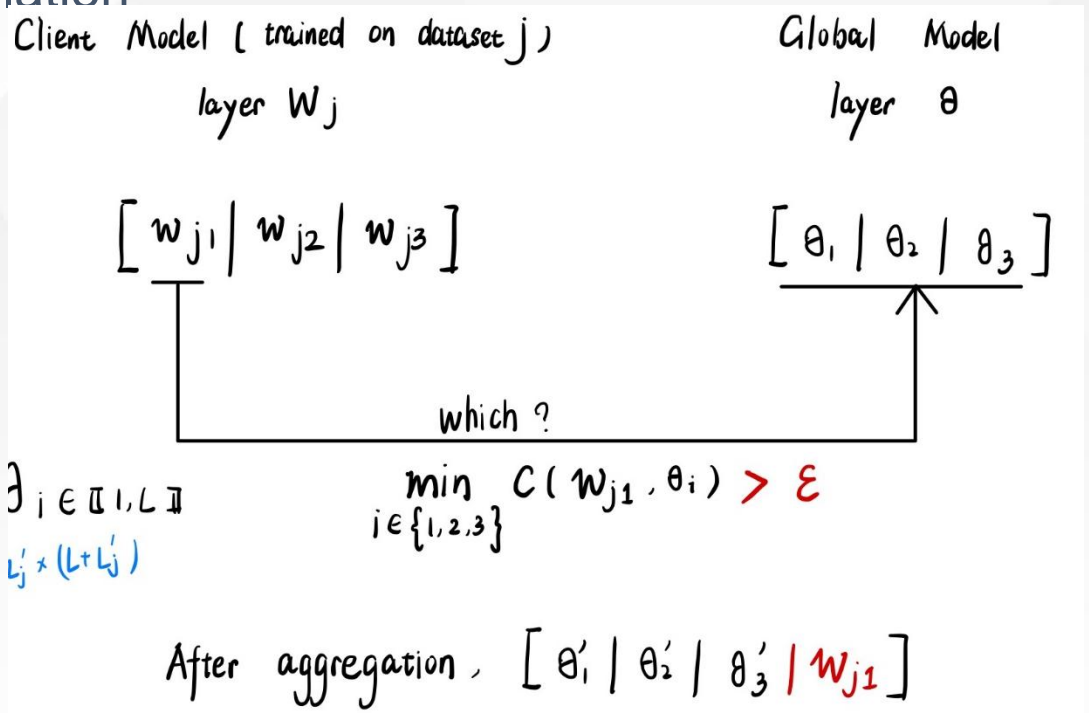
- Matched averaging formulation



$$\min_{\{\pi_{li}^j\}} \sum_{i=1}^{L} \sum_{j,l} \min_{\theta_i} \pi_{li}^j c(w_{jl}, \theta_i) \text{ s.t. } \sum_i \pi_{li}^j = 1 \; \forall \, j, l; \; \sum_l \pi_{li}^j = 1 \; \forall \, i, j.$$

FedMA algorithm

- Matched averaging formulation

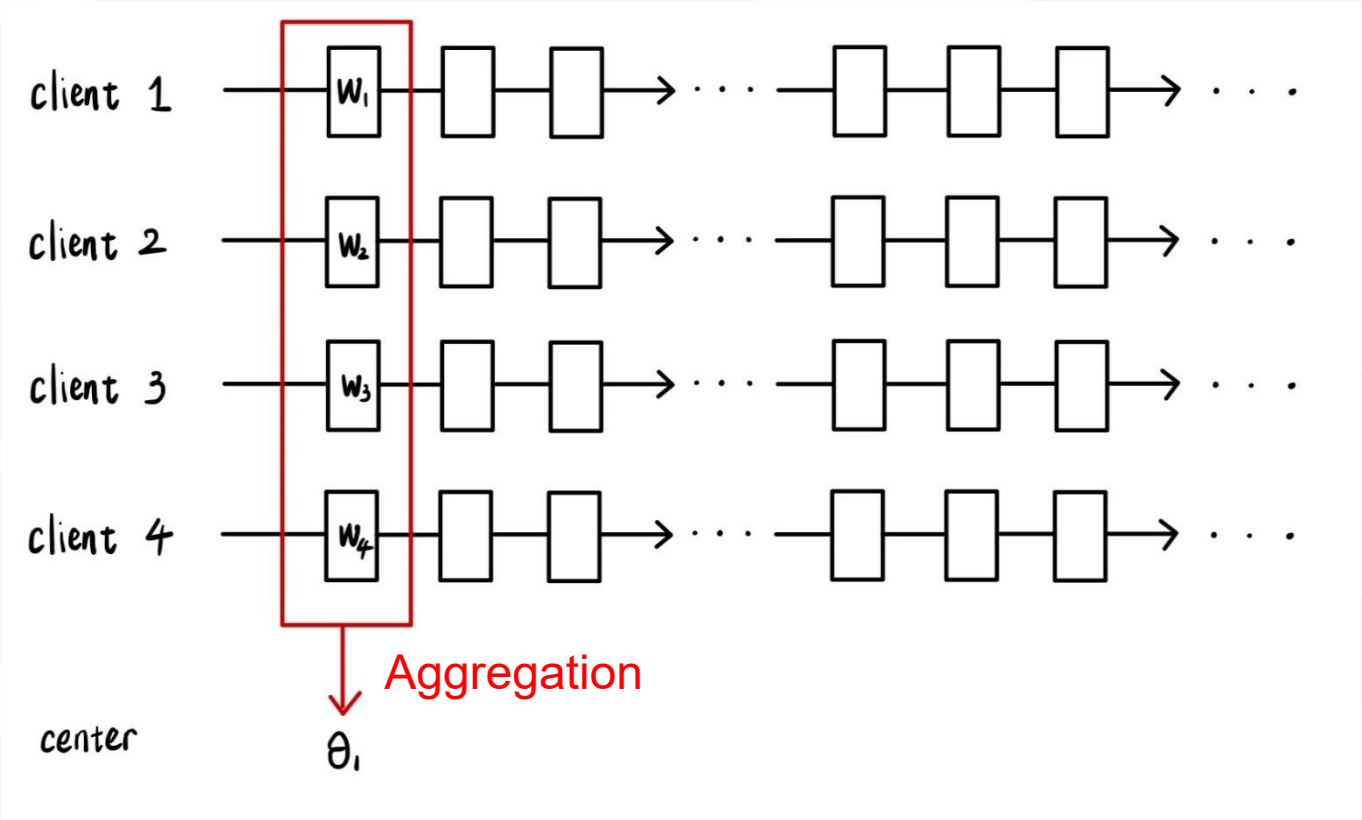


$$\min_{\{\pi_{li}^{j'}\}_{l,i}} \sum_{i=1}^{L+L_{j'}} \sum_{j=1}^{L_{j'}} \pi_{li}^{j'} C_{li}^{j'} \text{ s.t. } \sum_{i} \pi_{li}^{j'} = 1 \; \forall\, l; \; \sum_{l} \pi_{li}^{j} \in \{0,1\} \; \forall\, i, \text{ where}$$

$$C_{li}^{j'} = \begin{cases} c(w_{j'l}, \theta_i), & i \leq L \\ \epsilon + f(i), & L < i \leq L + L_{j'}. \end{cases}$$

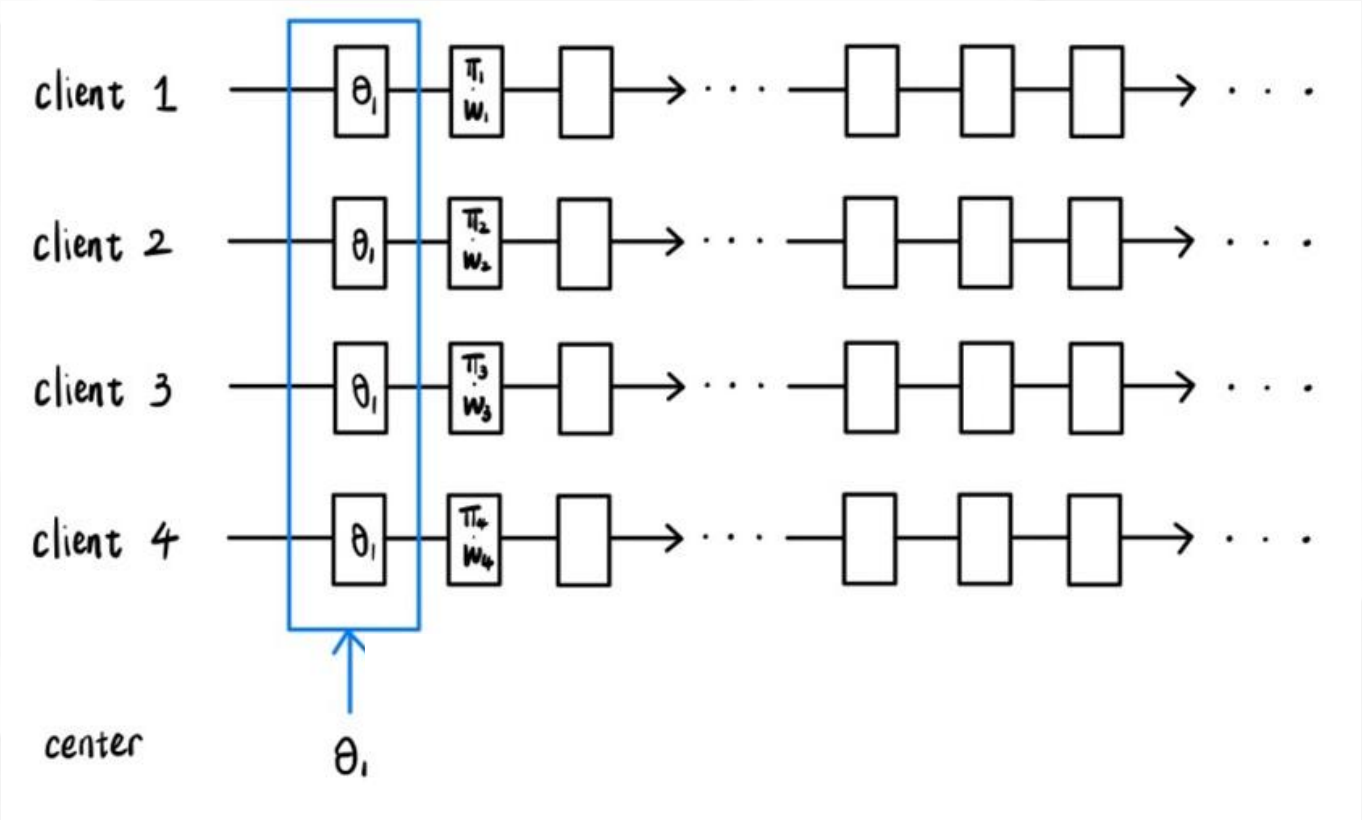
FedMA algorithm

- FedMA (https://github.com/IBM/FedMA)



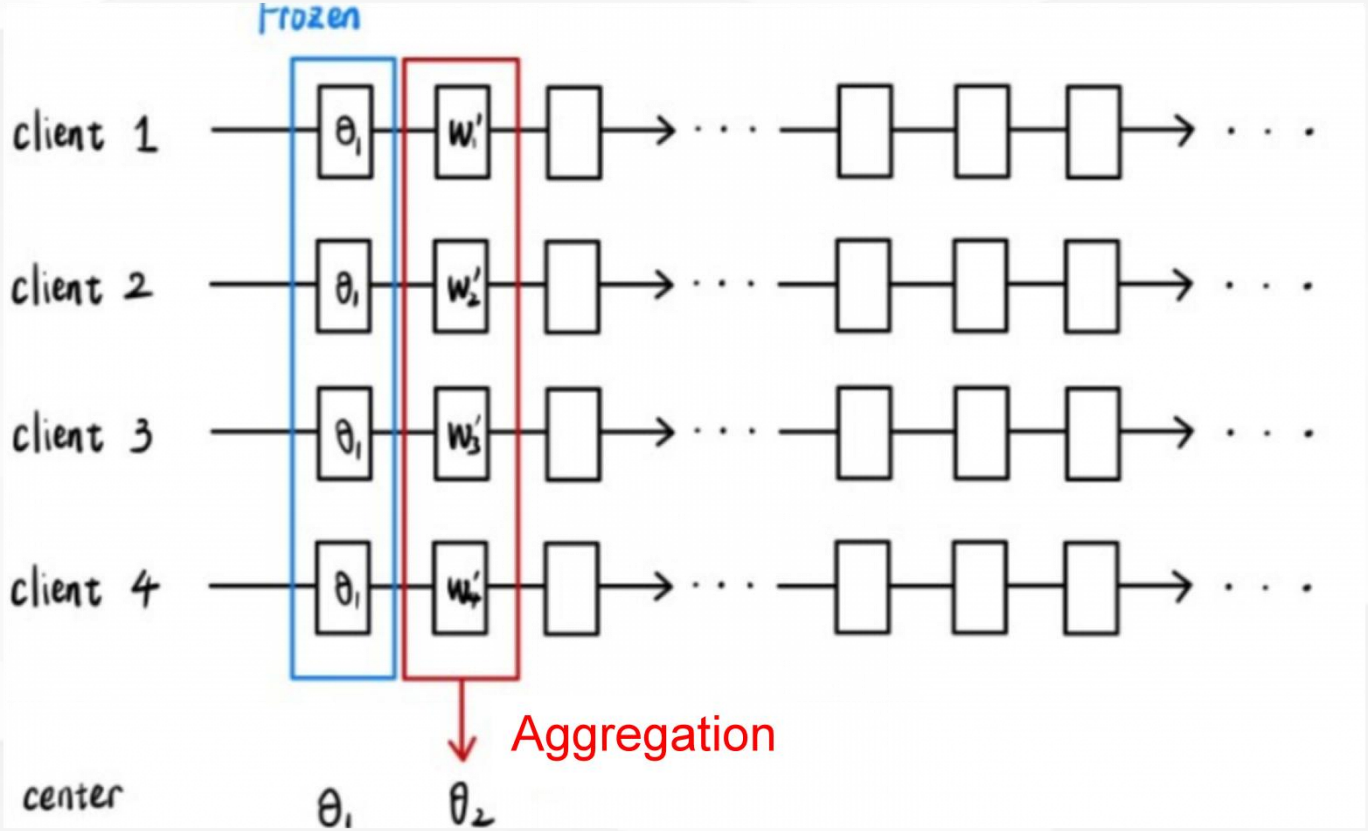
FedMA algorithm

- FedMA (https://github.com/IBM/FedMA)



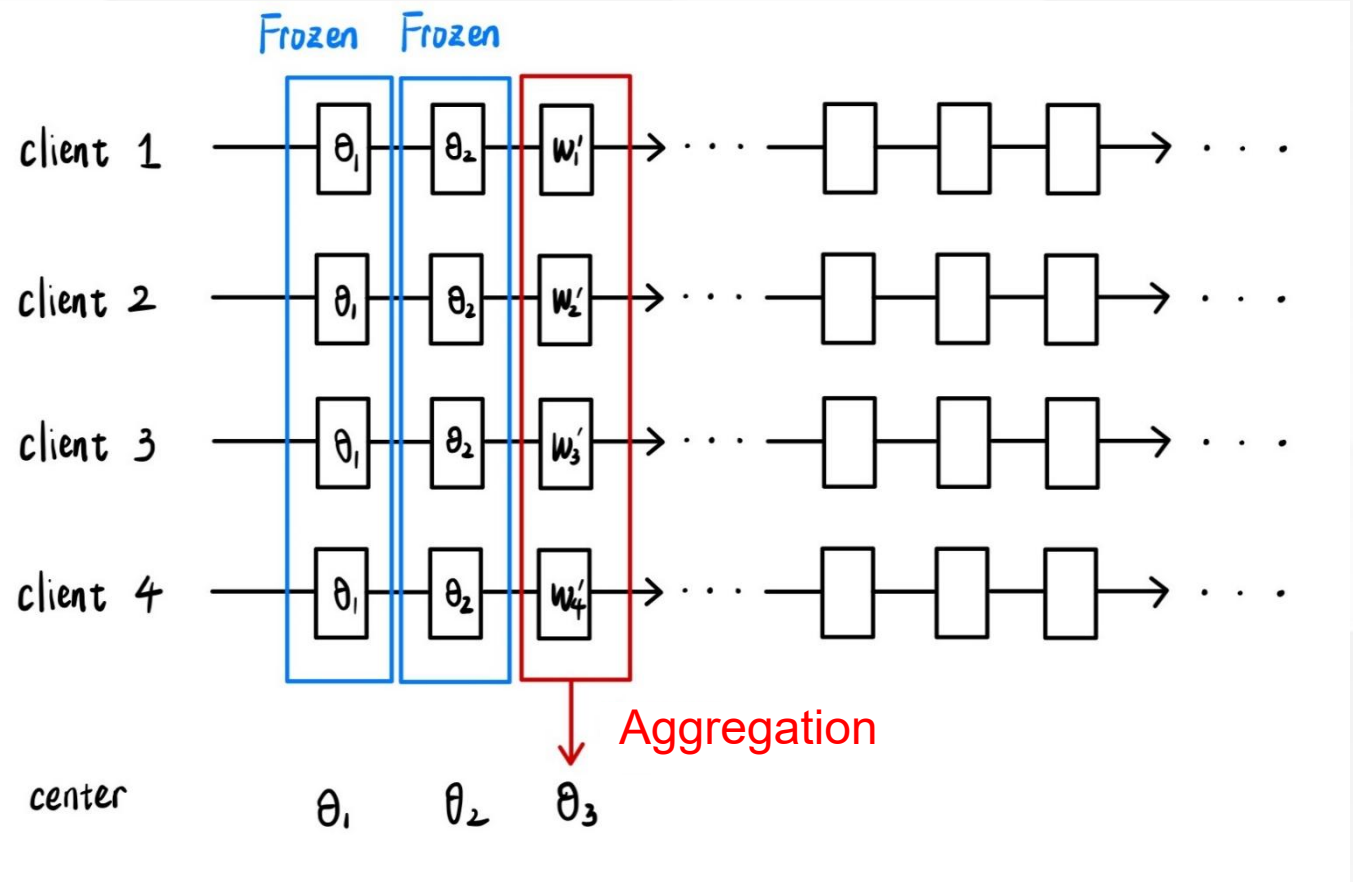
FedMA algorithm

- FedMA (https://github.com/IBM/FedMA)



FedMA algorithm

- FedMA (https://github.com/IBM/FedMA)



FedMA algorithm

- FedMA (https://github.com/IBM/FedMA)

Algorithm 1: Federated Matched Averaging (FedMA)

Input : local weights of N-layer architectures $\{W_{j,1}, \ldots, W_{j,N}\}_{j=1}^{J}$ from J clients

Output: global weights $\{W_1, \ldots, W_N\}$

$n = 1$;

while $n \leq N$ **do**

  if $n < N$ **then**

    $\{\Pi_j\}_{j=1}^{J} = \text{BBP-MAP}(\{W_{j,n}\}_{j=1}^{J})$; // call BBP-MAP to solve Eq. 2

    $W_n = \frac{1}{J} \sum_j W_{j,n} \Pi_j^T$;

  else

    $W_n = \sum_{k=1}^{K} \sum_j p_{jk} W_{jl,n}$ where p_k is fraction of data points with label k on worker j;

  end

  for $j \in \{1, \ldots, J\}$ **do**

    $W_{j,n+1} \leftarrow \Pi_j W_{j,n+1}$; // permutate the next-layer weights

    Train $\{W_{j,n+1}, \ldots, W_{j,L}\}$ with W_n frozen;

  end

  $n = n + 1$;

end

03

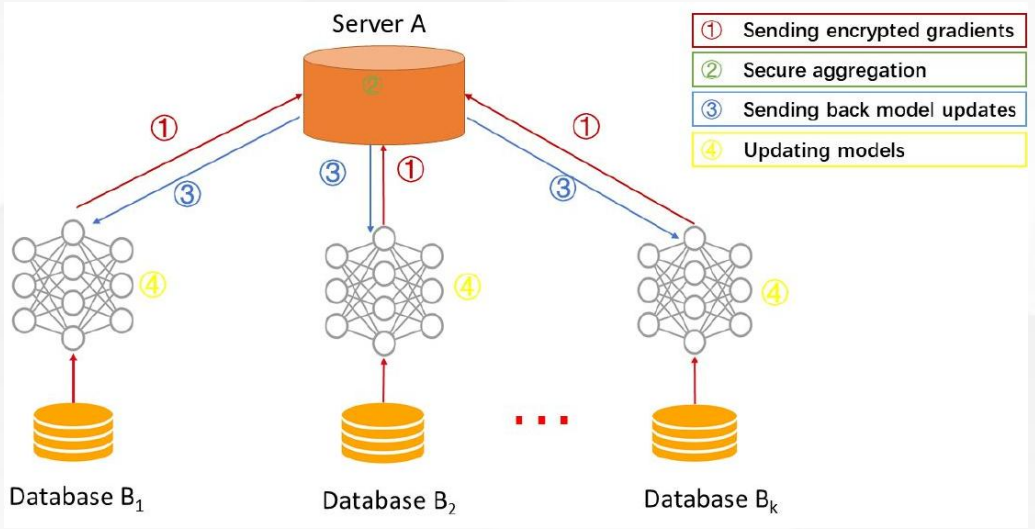
Advanced algorithms

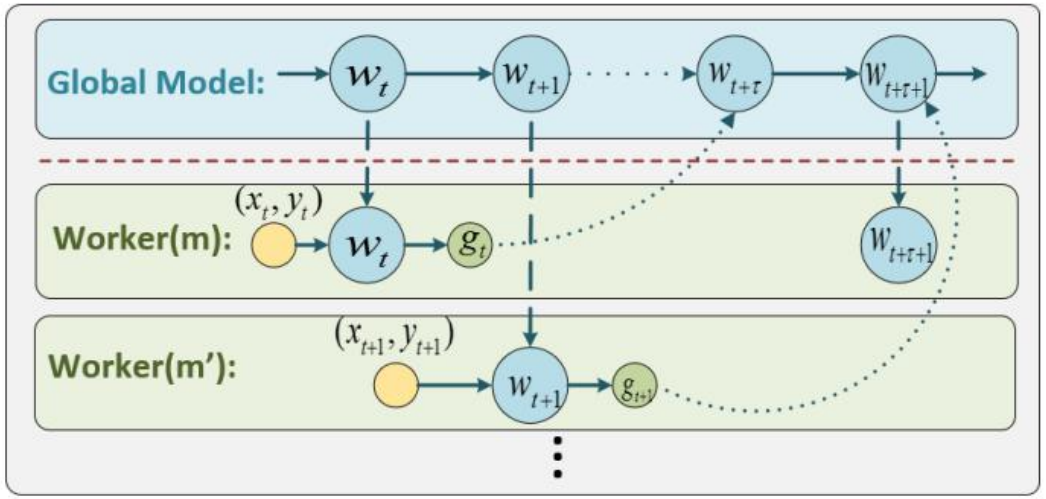
- SageFlow
- FLChain
- FAIR

SageFlow

Sageflow algorithm

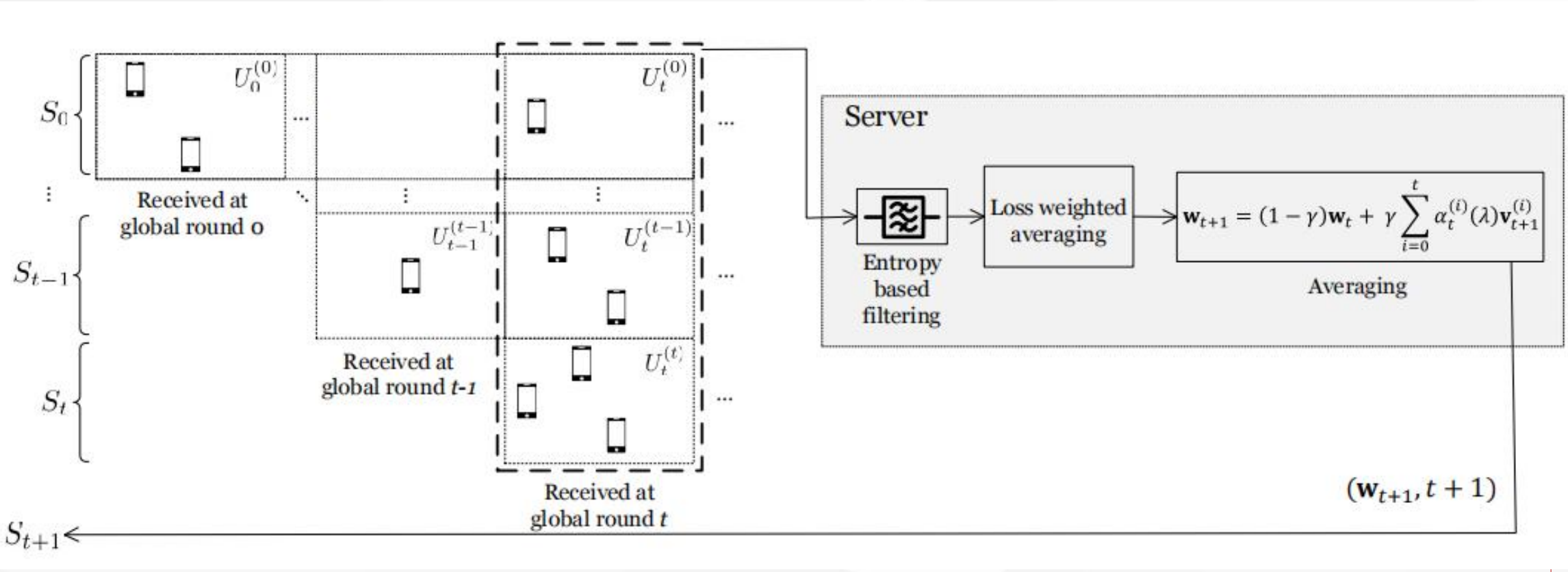
- Stragglers: slow devices
  - **Keep waiting**: slow down the overall process
  - **Drop out**: important data missing
  - **Asynchronous**(staleness): + adversaries?
- Attackers: malicious attacks launched by adversaries
  - **untargeted attacks**: model poisoning, data poisoning
  - **targeted/backdoor attacks**: misclassify the targeted subtasks
  - Robust Federated Averaging & Multi-Krum
- Large portion of adversaries
  - Straggler: increase attack ratio





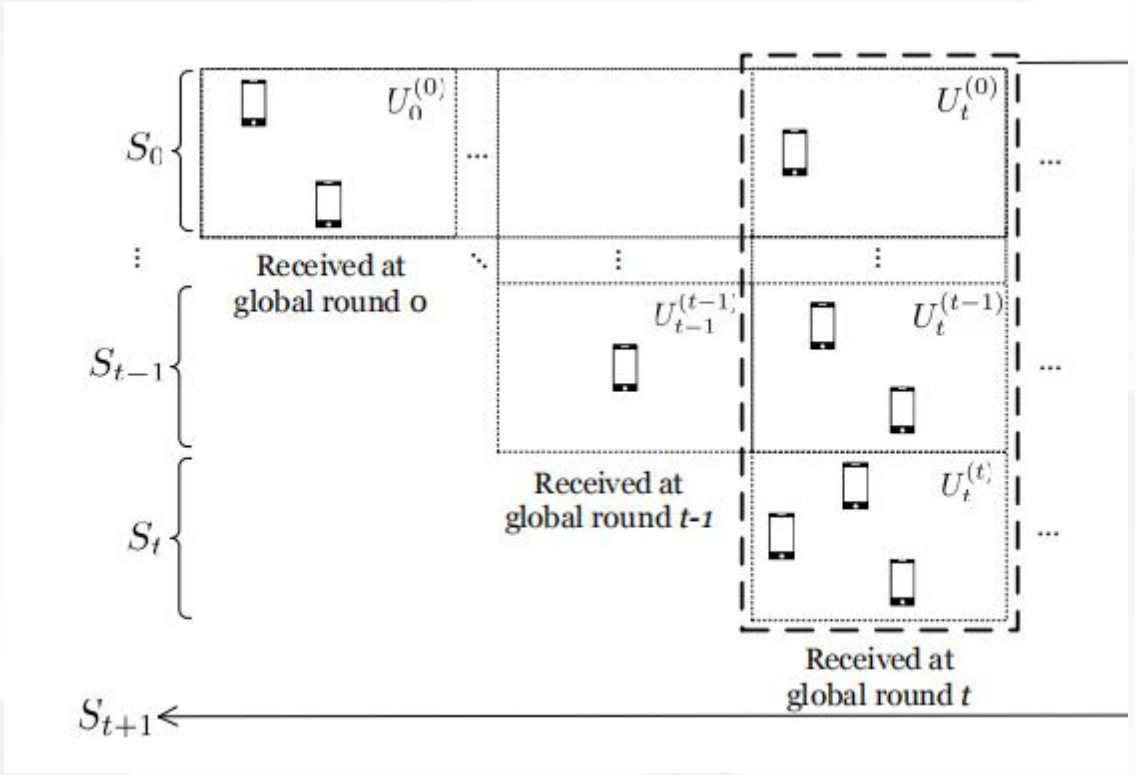
Sageflow algorithm

- Staleness-aware grouping
- Entropy-based filtering + Loss-weighted averaging



Sageflow algorithm

- Staleness-aware grouping
  - perform **periodic** global aggregation(fixed time deadline)
  - allow stragglers to be aggregated in **later rounds**
  - group with **same staleness** -> group representative model
  - aggregate according to **staleness**



Staleness group Number of data samples

$$\mathbf{v}_{t+1}^{(i)} = \sum_{k \in U_t^{(i)}} \frac{m_k}{\sum_{k \in U_t^{(i)}} m_k} \mathbf{w}_i(k)$$

$$\mathbf{w}_{t+1} = (1-\gamma)\mathbf{w}_t + \gamma \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}$$

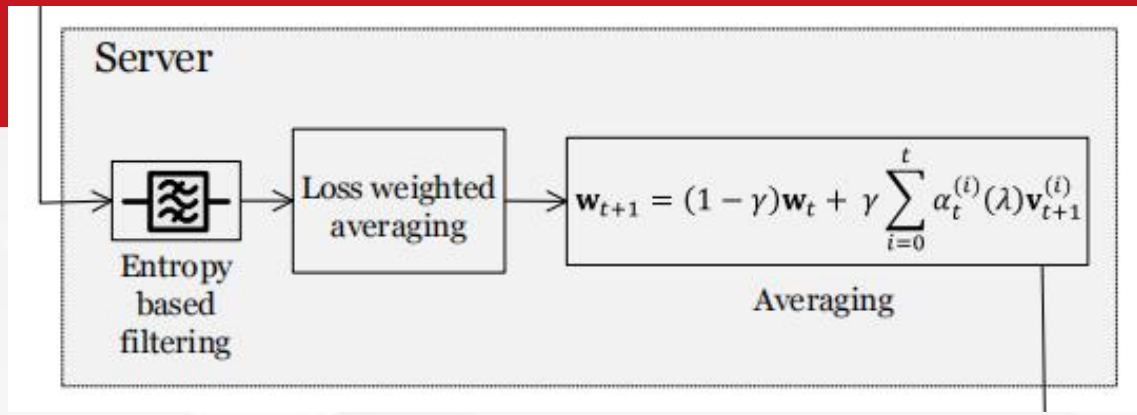
$$\sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}$$

$$\alpha_t^{(i)}(\lambda) \propto \frac{\sum_{k \in U_t^{(i)}} m_k}{(t-i+1)^{\lambda}}$$

Staleness function

Sageflow algorithm

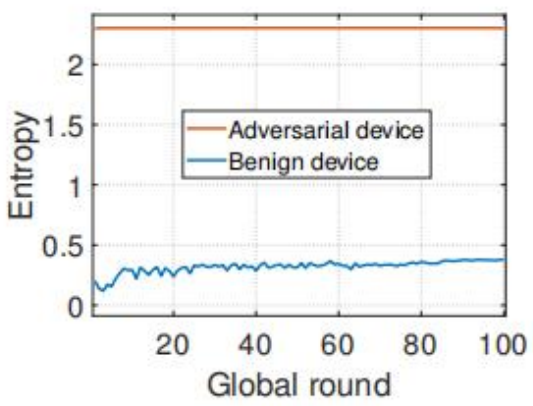
- Entropy-based filtering
  - **Public data** on server
  - **Filter out** high **entropy** models (loss)
  - For model poisoning



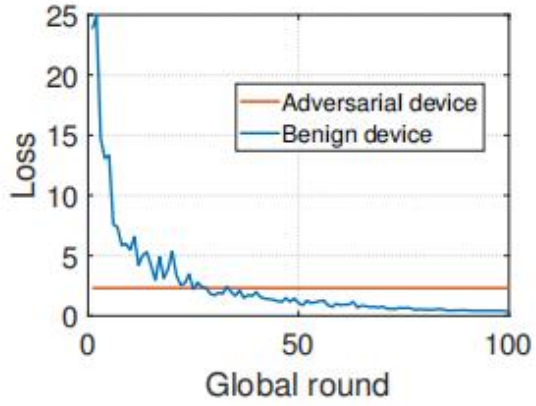
$$E_{x_{pub,j}}(k) = -\sum_{q=1}^{Q} P^{(q)}_{x_{pub,j}}(k) \log P^{(q)}_{x_{pub,j}}(k)$$

$$E(k) = \frac{1}{n_{pub}} \sum_{j=1}^{n_{pub}} E_{x_{pub,j}}(k)$$

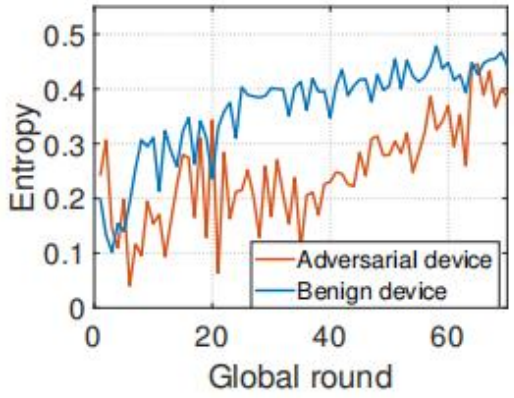
Shannon entropy



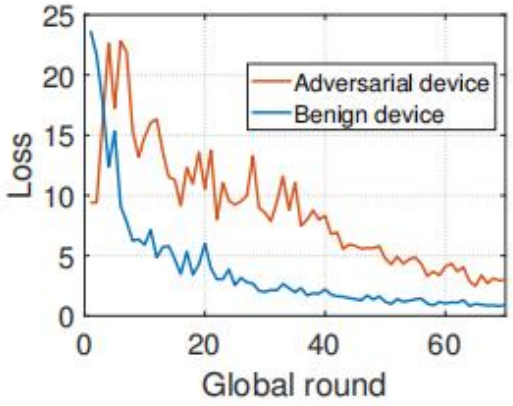
(a) Model poison, entropy



(b) Model poison, loss



(c) Data poison, entropy



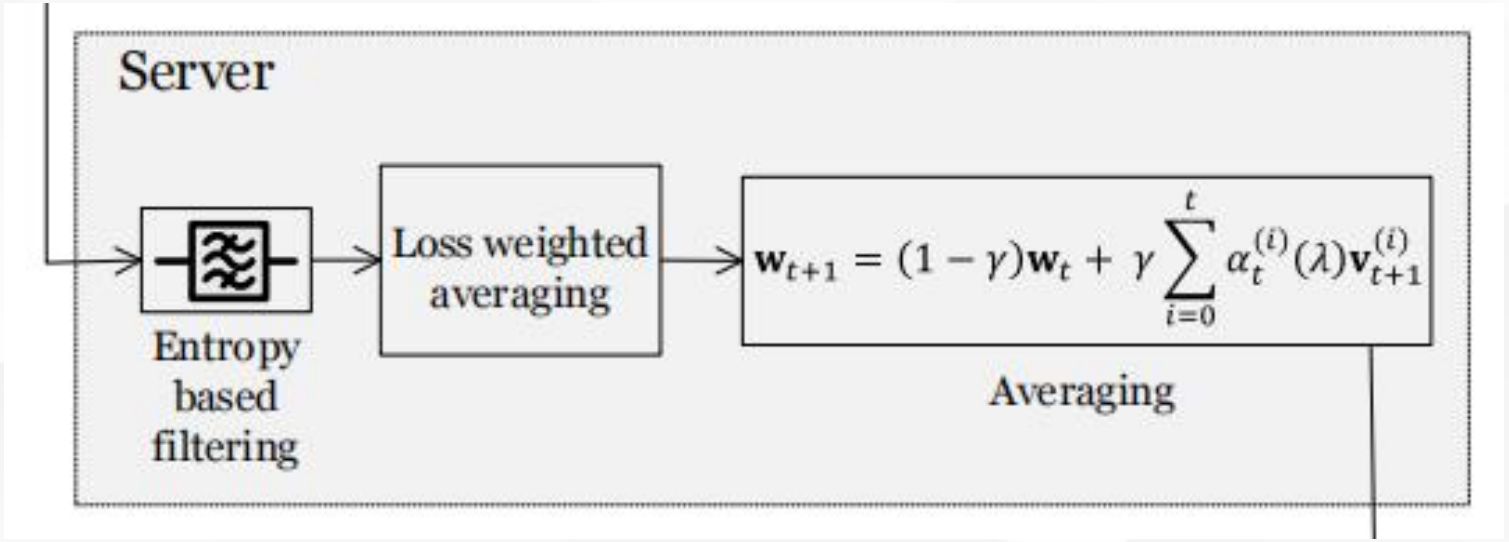
(d) Data poison, loss

Sageflow algorithm

- Loss-weighted averaging
  - Aggregation weight according to local models' **measured qualities**
  - Measure by **loss** on public data
  - data-poisoned model -> small weight + less impact
  - For data poisoning & scaled backdoor

$$\beta_t^{(k)}(\delta) \propto \frac{m_k}{\{F_{pub}(\mathbf{w}_t(k))\}^{\delta}} \quad \text{and} \quad \sum_{k \in S_t} \beta_t^{(k)}(\delta) = 1.$$

$$\mathbf{w}_{t+1} = \sum_{k \in S_t} \beta_t^{(k)}(\delta) \mathbf{w}_t(k)$$



Sageflow algorithm

- Time complexity

Public data: n_{pub} | Model parameters: $|w|$ | Clients number: K

$$\mathcal{O}(n_{pub}|w|K)$$

Algorithm 1 Proposed Sageflow Algorithm

Input: Initialized model $\mathbf{w}_0$, **Output:** Final global model $\mathbf{w}_T$

Process at the Server

1: **for** each global round $t = 0, 1, ..., T-1$ **do**
2: Choose S_t and send the current model and the global round $(\mathbf{w}_t, t)$ to the devices
3: Wait for T_d and then:
4: **for** $i = 0, 1, ..., t$ **do**
5: $U_t^{(i)}(E_{th}) = \{k \in U_t^{(i)} | E(k) < E_{th}\}$ // Entropy-based filtering in each group
6: $\mathbf{v}_{t+1}^{(i)} = \sum_{k \in U_t^{(i)}(E_{th})} \beta_t^{(k)}(\delta)\mathbf{w}_i(k)$ // Loss-weighted averaging in each group (with same staleness)
7: **end for**
8: $\mathbf{w}_{t+1} = (1-\gamma)\mathbf{w}_t + \gamma \sum_{i=0}^{t} \alpha_t^{(i)}(\lambda)\mathbf{v}_{t+1}^{(i)}$ // Averaging of representative models (with different staleness)
9: **end for**

Process at the Device: Device k receives $(\mathbf{w}_t, t)$ from the server and performs local updates to obtain $\mathbf{w}_t(k)$. Then each benign device k sends $(\mathbf{w}_t(k), t)$ to the server, while a malicious adversary sends a poisoned model depending on the type of attack.

Sageflow algorithm

- Theoretical Analysis
- Convergence analysis
  - Assumption 1: μ-strongly convex + L-smooth
  - Assumption 2: unbiased estimation
- Theoretical bound

$$F(x) \leq F(y) + \nabla F(x)^T (x-y) - \frac{\mu}{2}\|x-y\|^2$$

$$F(x) \geq F(y) + \nabla F(x)^T (x-y) - \frac{L}{2}\|x-y\|^2$$

$$\mathbb{E}\|\nabla F_k(\mathbf{w}_t^i(k), \xi_t^i(k)) - \nabla F(\mathbf{w}_t^i(k))\|^2 \leq \rho_1$$

$$\mathbb{E}[F(\mathbf{w}_T) - F(\mathbf{w}^*)] \leq \nu^T [F(\mathbf{w}_0) - F(\mathbf{w}^*)] + (1-\nu^T) Z(\lambda, E_{th}, \delta)$$

Convergence speed

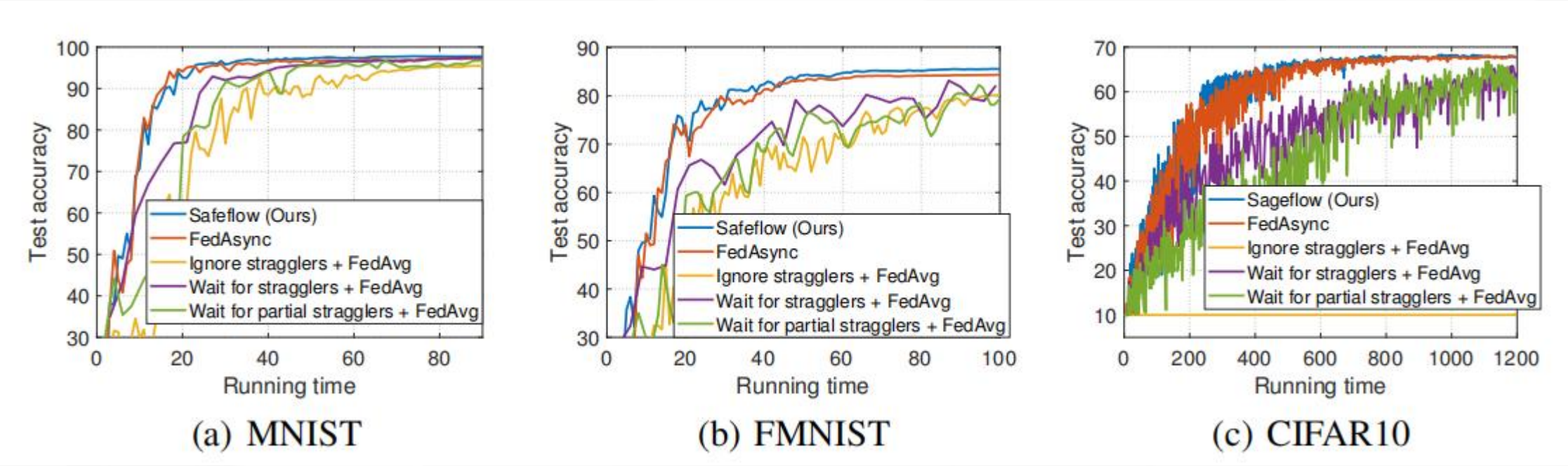
Error

Sageflow algorithm

- Datasets: MNIST, FMNIST, CIFAR10
  - 2% as public data
- Models: CNN(2conv+2fc), CNN(2conv+1fc), CNN(VGG-11)
  - ignore batchnorm
- FL setting: 100 clients, two classes for each client, 5 local epochs, batch size of 10

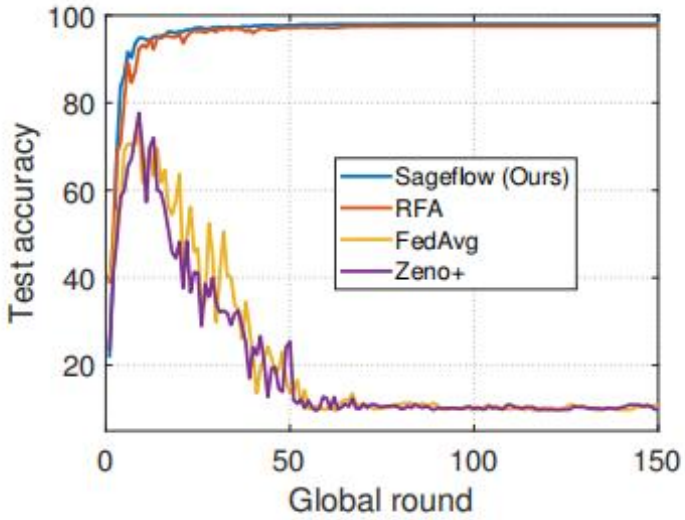
Sageflow algorithm

- Only stragglers: 10% participants
- Baselines: FedAvg(waiting, ignoring, waiting 50%), FedAsync
- Settings: uniform delay of [0,1,2] global rounds
- Ignoring lose significant data converges to a **suboptimal point**
- Waiting(all, 50%) requires the **largest running time** until convergence

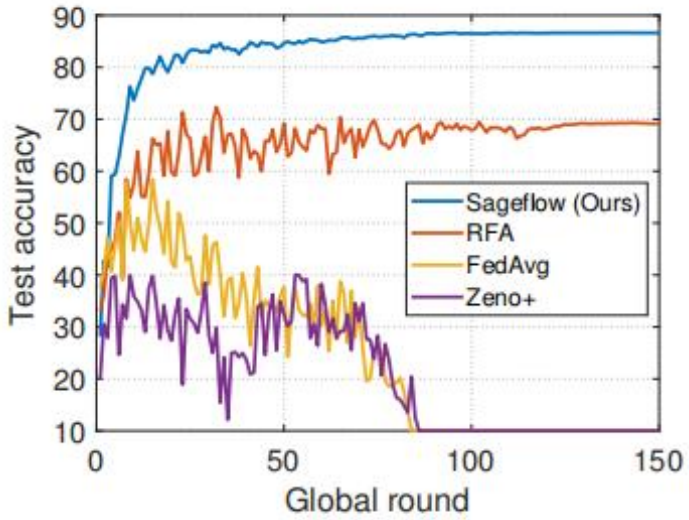


Sageflow algorithm

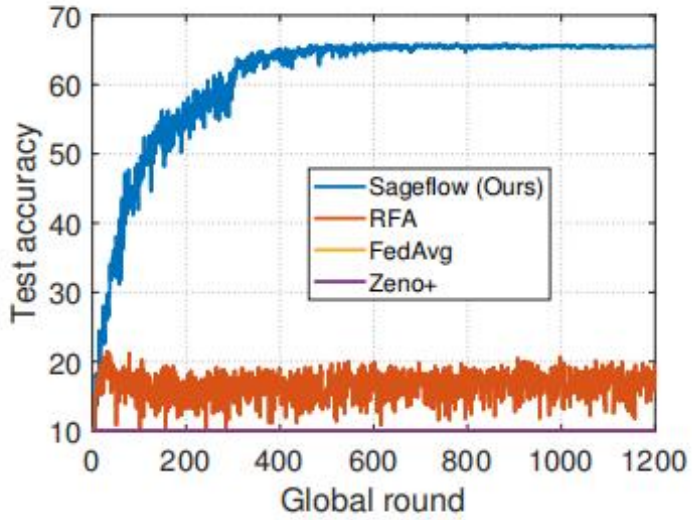
- Only adversaries: 20% participants
- Baselines: RFA, FedAvg, synchronized Zeno+, Multi-Krum
- Attacks: model(-0.1w), data(label-flipping), backdoor(model replacement, pixel-pattern attack)
- FedAvg **does not work well** on all datasets
- RFA: **complex model** led to worse performance
- Zeno+: bad on poisoning **but** good for backdoor
- Sageflow: **slow down** posioning



(a) MNIST



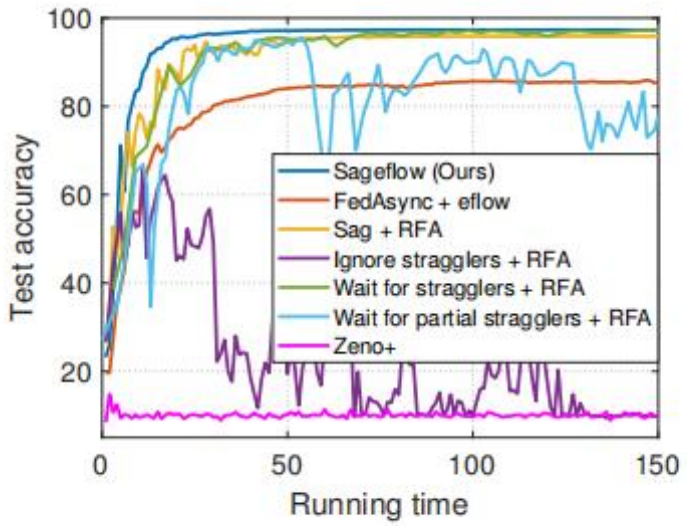
(b) FMNIST



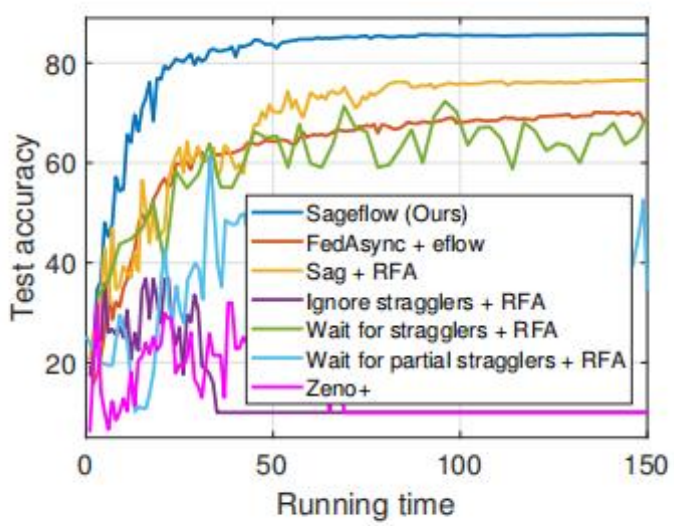
(c) CIFAR10

Sageflow algorithm

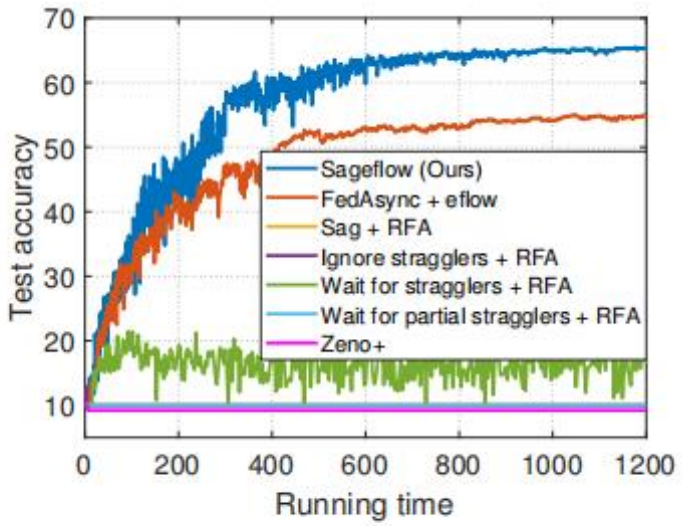
- Stragglers + adversaries: 20%(model/data), 10%(backdoor) participants
- Baselines: asynchronized Zeno+, Multi-Krum
- Zeno+: **does not perform well**(ignore staleness & entropy)
- Waiting + RFA: suffer from **straggler**
- Ignoring/Sag + RFA: poor(**high attack ratio**)
- eflow/RFA + FedAsync: poor(**one-by-one arrivals**)



(a) MNIST



(b) FMNIST



(c) CIFAR10

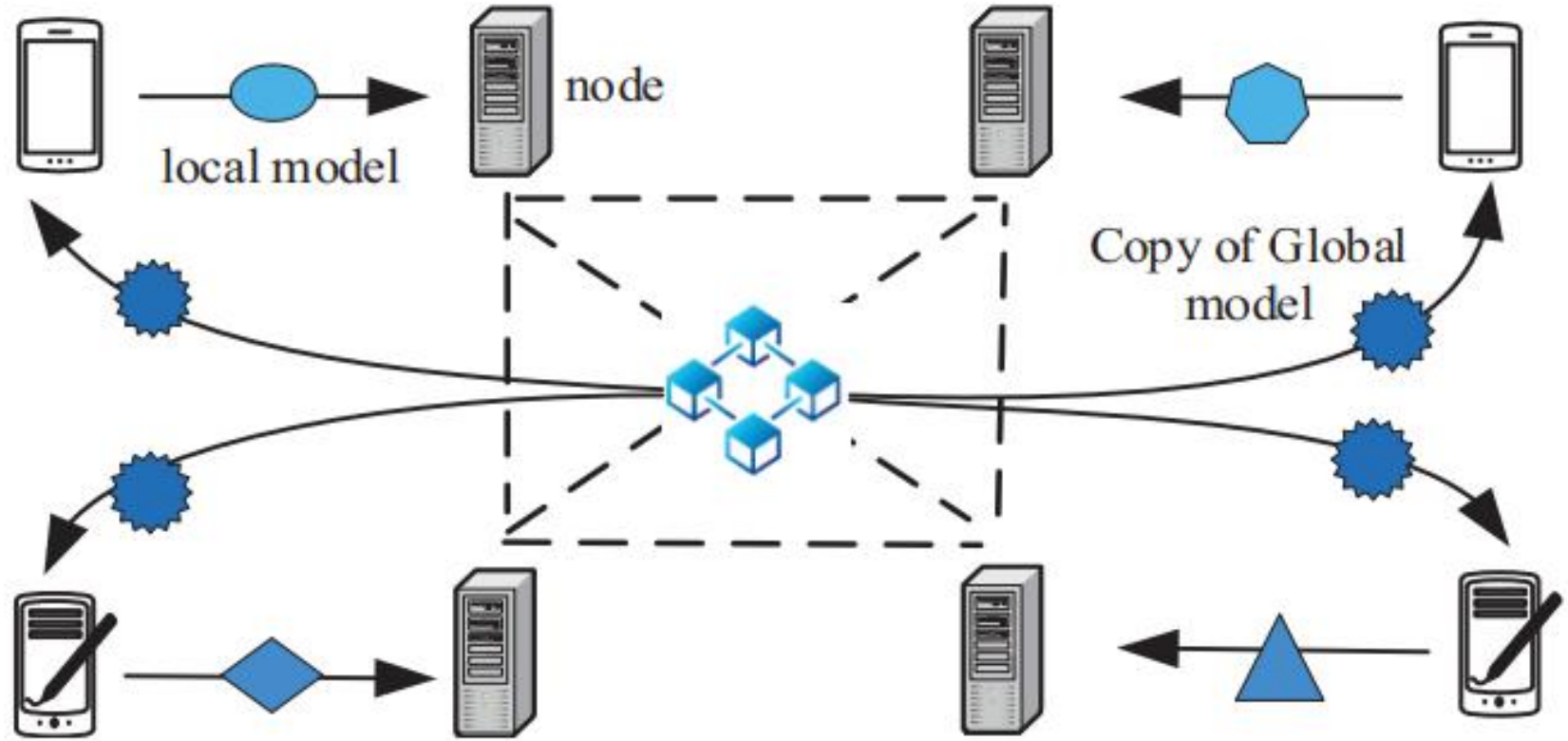
Sageflow algorithm

- Sageflow: robust FL scheme handle both **stragglers and adversaries**
  - staleness-aware grouping: stragglers
  - entropy-based filtering: model poisoning
  - loss-weighted averaging: data poisoning + backdoor
- Theoretical convergence analysis
- Extensive experimental results
- Future issues: Sageflow + secure aggregation

FLChain

FL + Blockchain

- complete dependency on the reliability of a central server for storage and computation of the global model update
- Any malicious activity leads to flawed global model update which is detrimental for accuracy of subsequent local model updates



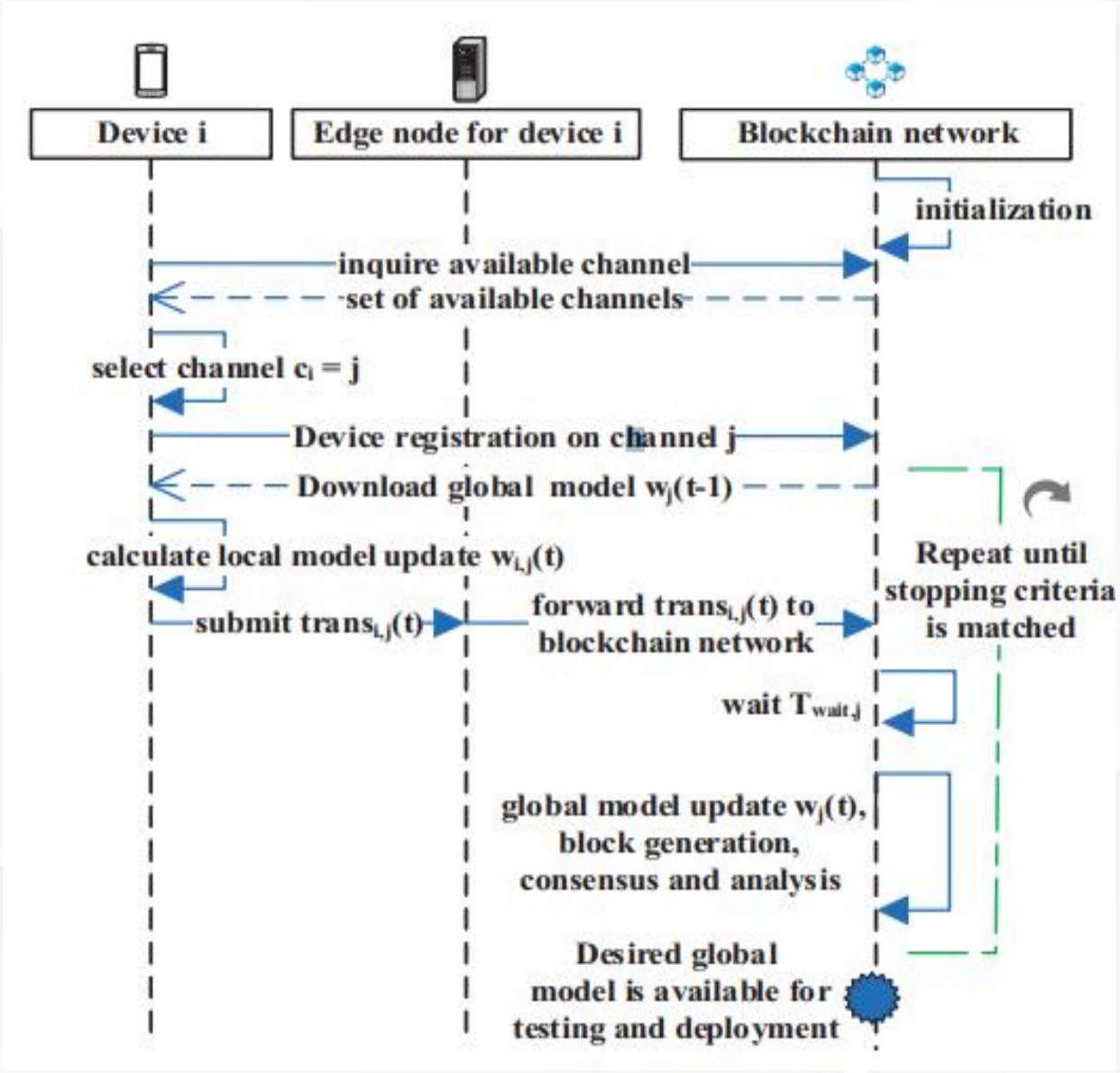
FL + Blockchain

- Initialization
- Channel Inquiry
- Channel Selection
- Device Registration
- Local Model Update
- Transaction Pool
- Global Model Update
- Consensus Protocol
- Analysis

Algorithm 1 : FLchain operation for channel j

1: Setup channel j for global model M_j
2: initialization: $t = 0$; $w_j(0), w_{i,j}(0) \in [0, w_{j,max}]$;
3: **for all** $i \in D_j$ **do in parallel**
4: Inquire available channels
5: Select channel $c_i = j \in C \triangleq \{1, 2, 3, \ldots, C_n\}$
6: Register device i to channel j
7: **end for**
8: **while** $\|w_j(t) - w_j(t-1)\|_2 \leq \varepsilon_{threshold,j}$ **do**
9: **for all** $i \in D_j$ **do in parallel**
10: Download $w_j(t)$ from channel j to device i
11: $t \leftarrow t + 1$; $w^0_{i,j}(t) = w_j(t)$;
12: **for** $v = 1, ..., V$ **do**
13: $w^v_{i,j}(t) = w^{v-1}_{i,j}(t) - \eta \nabla \Phi$, and Eq. (6)
14: **end for**
15: $w_{i,j}(t) = w^V_{i,j}(t)$, Generate $trans_{i,j}(t)$ and forward to blockchain network
16: Wait for notification from channel j
17: **end for**
18: Calculate $w_j(t)$ using Eq. (7)
19: global model state trie updation, block generation and consensus
20: **end while**

FL + Blockchain



04

Other algorithms

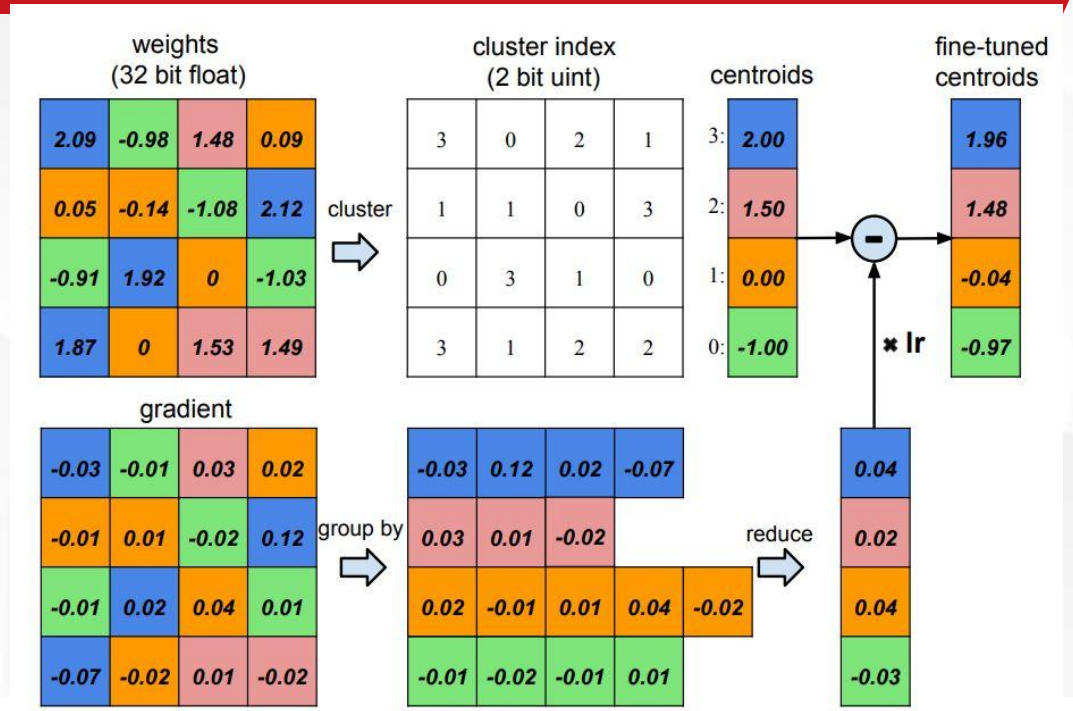
- 联邦学习效率
- 联邦学习安全
- 联邦学习异构性
- 其他论文

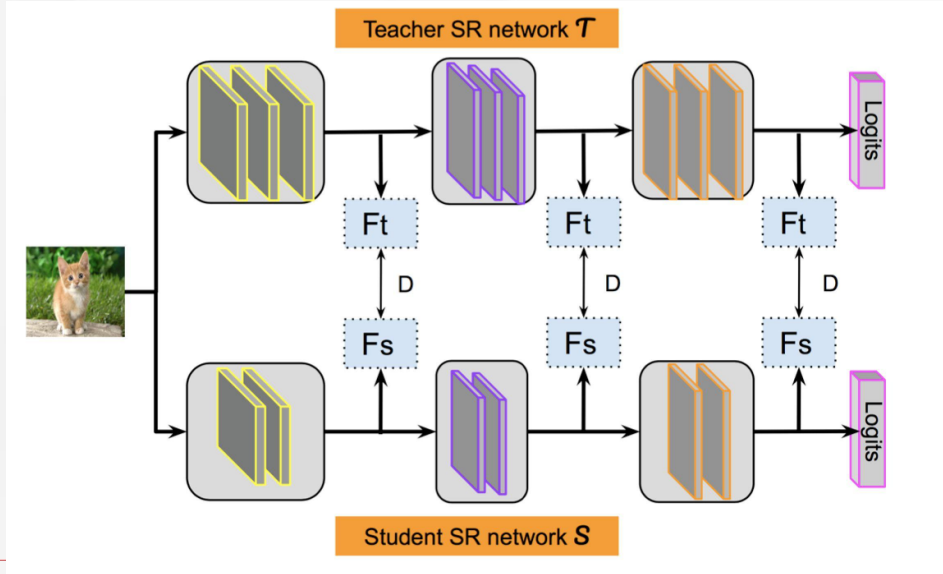
联邦学习效率

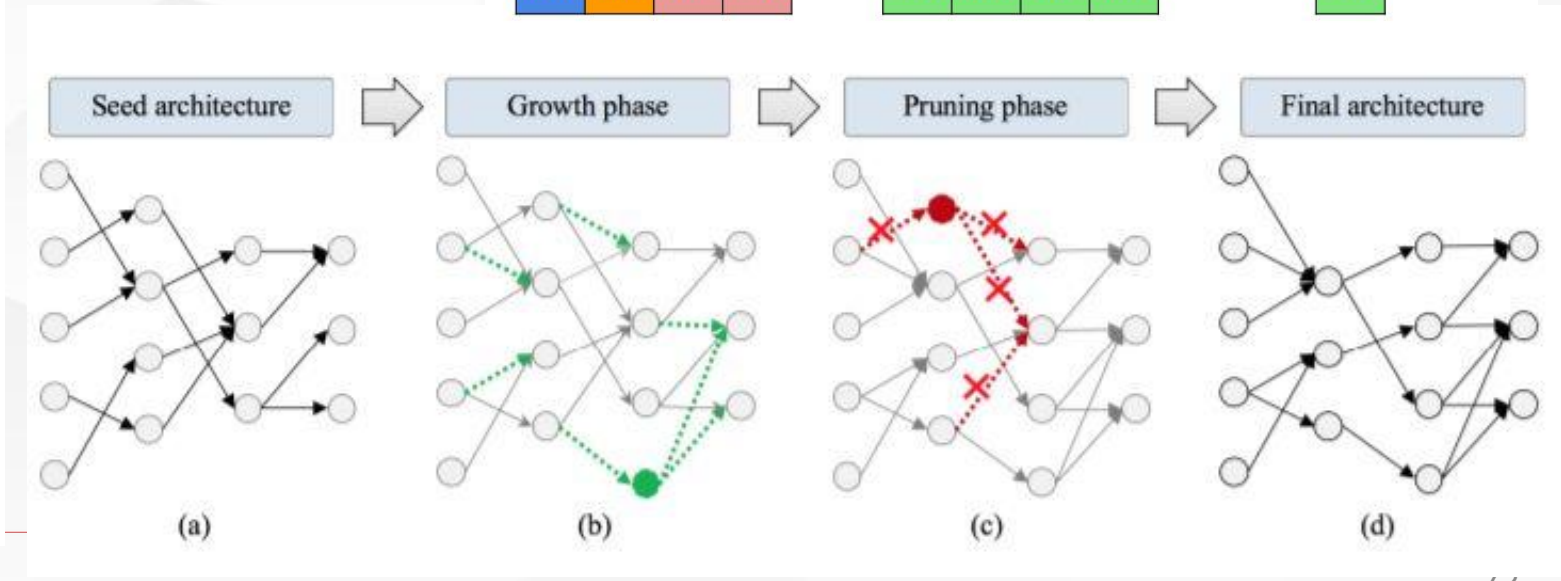
- 数据传输量大 - 模型压缩
- 网络环境复杂 - 传输调度

联邦学习效率-模型压缩

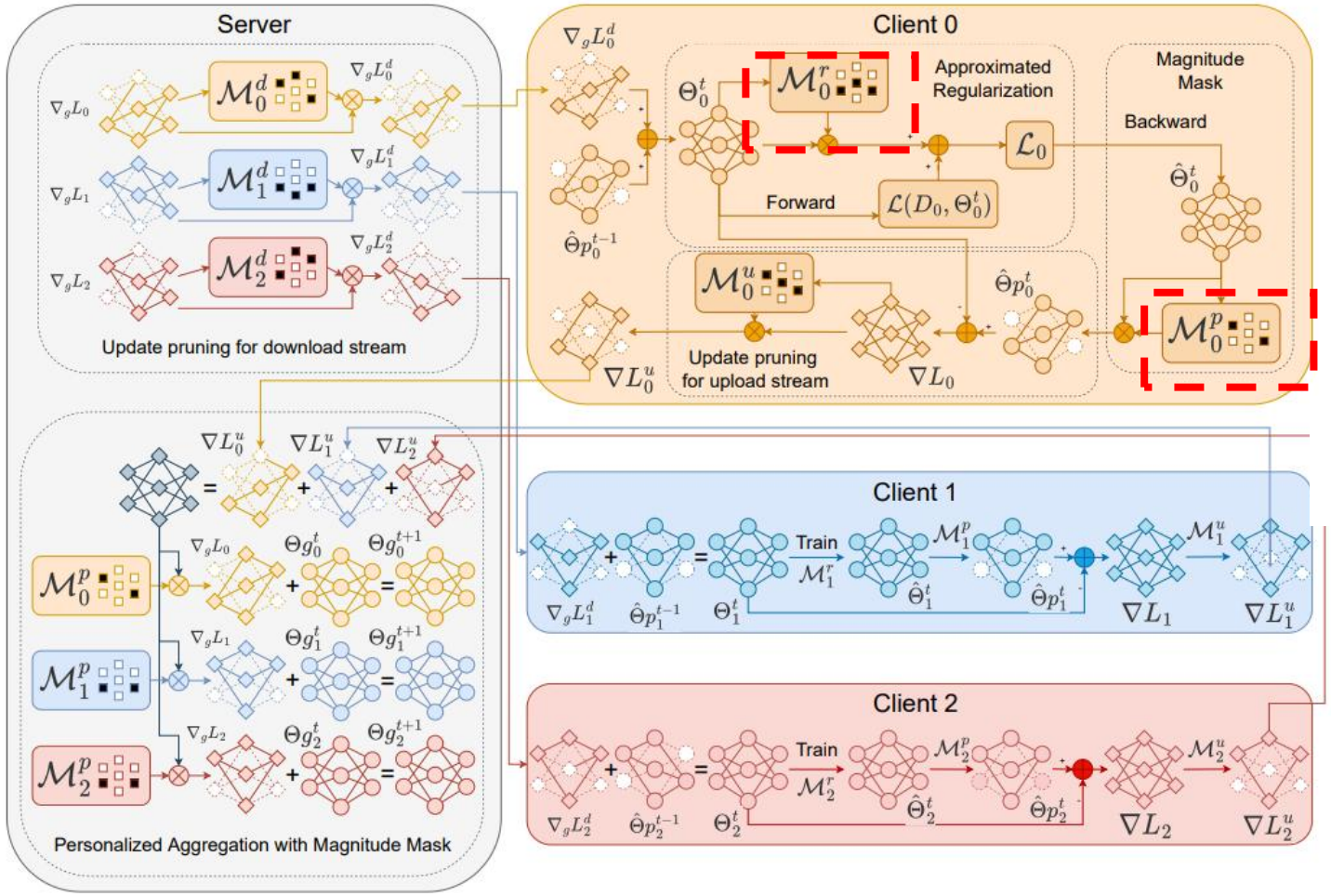
- 通信效率
  - 智能算法本身通信量大，通信开销大
- 模型压缩：**模型剪枝，模型量化，知识蒸馏**
  - 减少数据通信、计算量
  - 构造准确但小的神经网络模型







联邦学习效率



- 绝对值剪枝

$$\mathcal{H}(w) = |w|.$$

- 损失函数正则化
- 定位对推理结果影响较低的参数

$$\mathcal{L}_i = l(D_i, \Theta_i^t) + \lambda_1 \cdot ||\Theta_i^t \odot \mathcal{M}_i^r||_1.$$

- 训练后模型剪枝
- 过滤对推理结果影响较低的参数

$$\hat{\Theta}p_i^t \leftarrow \hat{\Theta}_i^t \odot \mathcal{M}_i^p.$$

Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

联邦学习效率

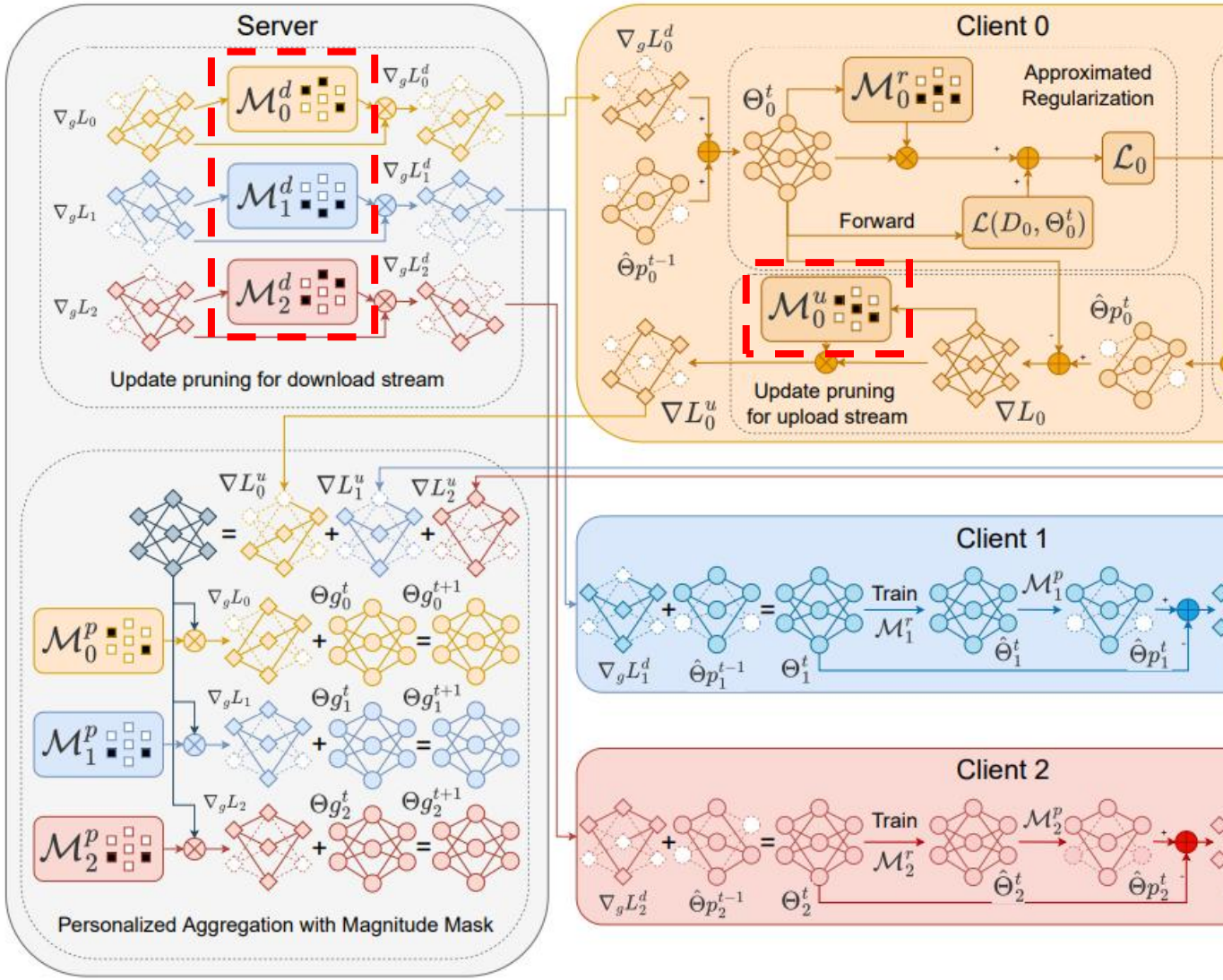
- 重要性剪枝

$$\begin{aligned}\mathcal{I}(w) &= |\mathcal{L}(D, \Theta_{w\to 0}) - \mathcal{L}(D, \Theta)| \\ &= |\frac{\partial \mathcal{L}(D,\Theta)}{\partial w} \cdot (0 - w) + o(w^2)| \\ &= |\frac{\partial \mathcal{L}(D,\Theta)}{\partial w} \cdot w|.\end{aligned}$$

- 上传/下载传输剪枝
- 过滤更新量较小的上传/下载模型更新

$$\nabla_g \mathcal{L}_i^u \leftarrow \nabla \mathcal{L}_i \odot \mathcal{M}_i^u.$$

$$\nabla \mathcal{L}_i^d = (\sum_{j=1}^{M} k_j \cdot \nabla_g \mathcal{L}_i) \odot \mathcal{M}_i^d$$



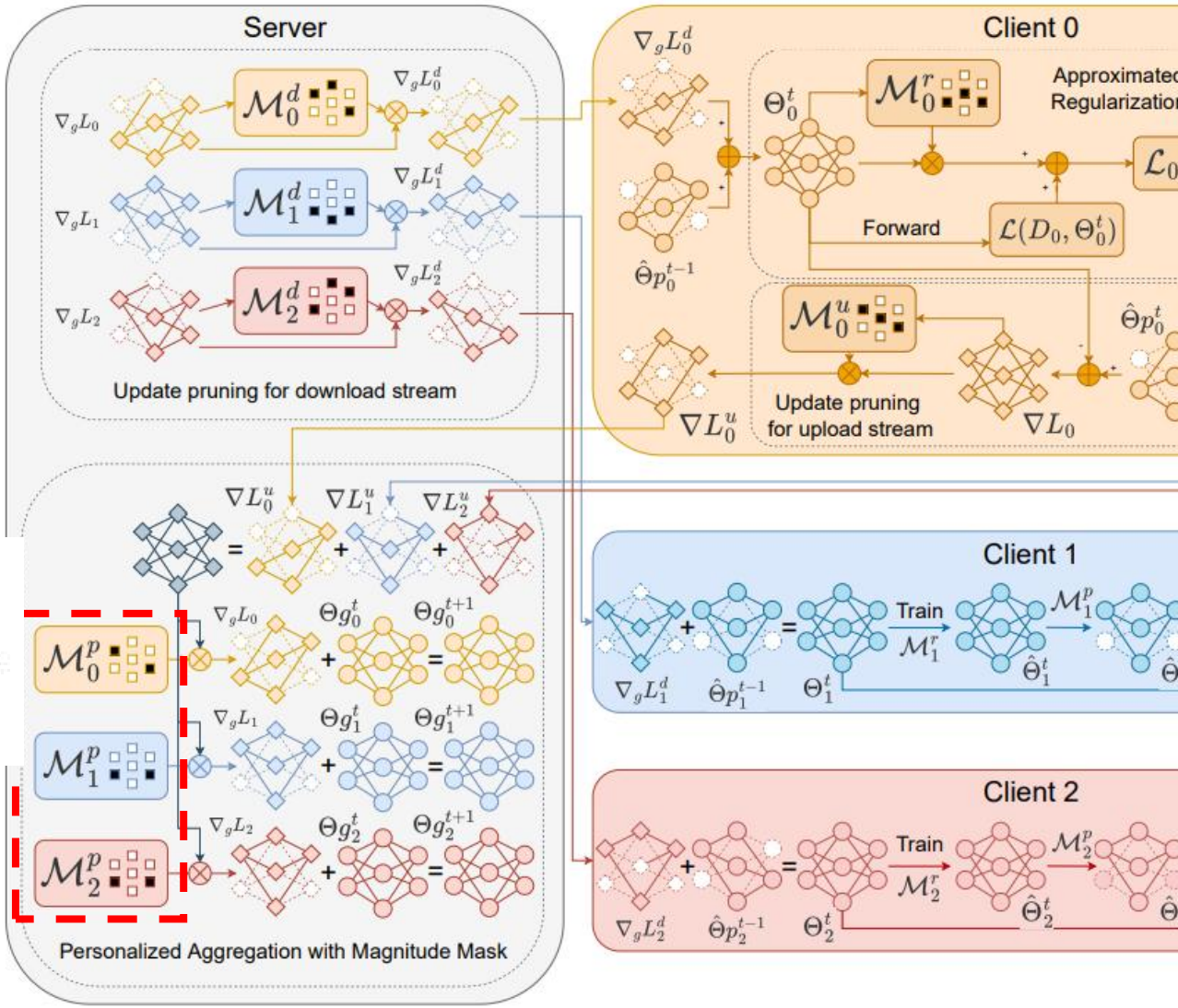
Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

联邦学习效率

- 个性化模型聚合
- 只对有效更新进行聚合

$$\Theta g_i^{t+1} \leftarrow \lambda_2 \cdot \Theta g_i^t +$$

$$(1-\lambda_2)\cdot \mathcal{M}_i^p \odot \sum_{j=1}^{M} k_j \cdot (\Theta g_j^t - \eta \cdot \nabla_g \mathcal{L}_j^u),$$



Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

联邦学习效率-模型压缩

- 在大幅减少通信量（~94%）的情况下
- 尽可能保障了训练准确率（<6%）

Metrics & Baselines			Dataset&Data distributions									
			MNIST					FMNIST				
			IID	DIR75	DIR50	DIR25	PAT	IID	DIR75	DIR50	DIR25	PAT
	Acc.	FedAvg	**0.9913**	0.9907	0.9881	0.9833	0.9576	0.9142	0.9021	0.9029	0.8923	0.7795
		FedPer	0.9687	0.9569	0.9512	0.9291	0.8778	0.8490	0.8373	0.8439	0.8230	0.7334
		ADP-50	0.9900	0.9899	0.9882	0.9848	0.9593	0.9016	0.9022	0.9047	0.8920	0.8006
		Hermes-50	0.9902	0.9899	0.9883	0.9852	0.9596	**0.9152**	0.9030	0.9031	0.8930	0.7803
		SplitFed	0.9911	**0.9909**	0.9882	0.9834	0.9545	0.9134	0.9022	0.9050	0.8919	0.7680
		FCCL	0.9820	0.9808	0.9821	**0.9866**	**0.9981**	0.8831	0.9037	0.9122	**0.9254**	**0.9949**
		LotteryFL-75	0.9700	0.9700	0.9750	0.9800	0.9775	**0.9152**	**0.9275**	**0.9125**	0.9250	0.9825
		LotteryFL-50	0.9700	0.9725	0.9700	0.9775	0.9875	0.8625	0.9075	0.9075	0.9225	0.9875
		LotteryFL-25	0.9350	0.9400	0.9350	0.9525	0.9600	0.7625	0.7675	0.7075	0.7850	0.8600
		pFedEff-75	0.9908	0.9873	**0.9888**	0.9814	0.9729	0.9108	0.8887	0.8961	0.8989	0.9293
		pFedEff-50	0.9897	0.9843	0.9862	0.9786	0.9715	0.9086	0.8774	0.8867	0.8894	0.9585
		pFedEff-25	0.9863	0.9723	0.9745	0.9601	0.9745	0.9018	0.8427	0.8544	0.8557	0.9500
	Conv.	FedAvg	198	198	192	197	192	196	200	200	198	200
		FedPer	196	197	199	200	197	196	196	199	197	200
		ADP-50	148	187	**179**	162	197	174	197	198	200	192
		Hermes-50	**90**	160	193	199	198	178	184	198	177	198
		SplitFed	192	198	198	189	199	196	189	194	190	193
		FCCL	79	187	167	132	184	151	127	191	189	51
		LotteryFL-75	131	178	198	**125**	98	**70**	109	93	152	41
		LotteryFL-50	197	**153**	183	168	149	159	182	157	165	81
		LotteryFL-25	187	189	181	159	172	199	**0**	**4**	**8**	97
		pFedEff-75	197	193	196	194	**2**	200	187	195	198	**4**
		pFedEff-50	199	199	195	193	4	190	191	199	196	6
		pFedEff-25	195	200	200	200	19	184	200	199	199	16
	Comm. Vol.	FedAvg	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		FedPer	0.0895	0.0895	0.0895	0.0895	0.0895	0.0895	0.0895	0.0895	0.0895	0.0895
		ADP-50	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		Hermes-50	1.0037	1.0037	1.0037	1.0037	1.0037	1.0037	1.0037	1.0037	1.0037	1.0037
		SplitFed	0.1071	0.1071	0.1071	0.1071	0.1071	0.1071	0.1071	0.1071	0.1071	0.1071
		FCCL	0.1031	0.1031	0.1031	0.1031	0.1031	0.1031	0.1031	0.1031	0.1031	0.1031
		LotteryFL-75	0.7350	0.7351	0.7351	0.7351	0.7350	0.7350	0.7351	0.7351	0.7351	0.7350
		LotteryFL-50	0.4876	0.4877	0.4877	0.4877	0.4876	0.4876	0.4877	0.4877	0.4877	0.4876
		LotteryFL-25	0.2377	0.2378	0.2378	0.2378	0.2377	0.3362	0.2430	0.2575	0.2523	0.2377
		pFedEff-75	0.5277	0.5625	0.5625	0.5625	0.5149	0.5548	0.5625	0.5625	0.5625	0.5623
		pFedEff-50	0.2398	0.2500	0.2500	0.2500	0.2407	0.2500	0.2500	0.2500	0.2500	0.2500
		pFedEff-25	**0.0596**	**0.0625**	**0.0625**	**0.0625**	**0.0625**	**0.0625**	**0.0625**	**0.0625**	**0.0625**	**0.0625**

Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

联邦学习效率-模型压缩

- 不同绝对值mask下不同压缩率结果
- pre-training mask和post-training mask能够有效减少通信量，但是对准确率影响较大

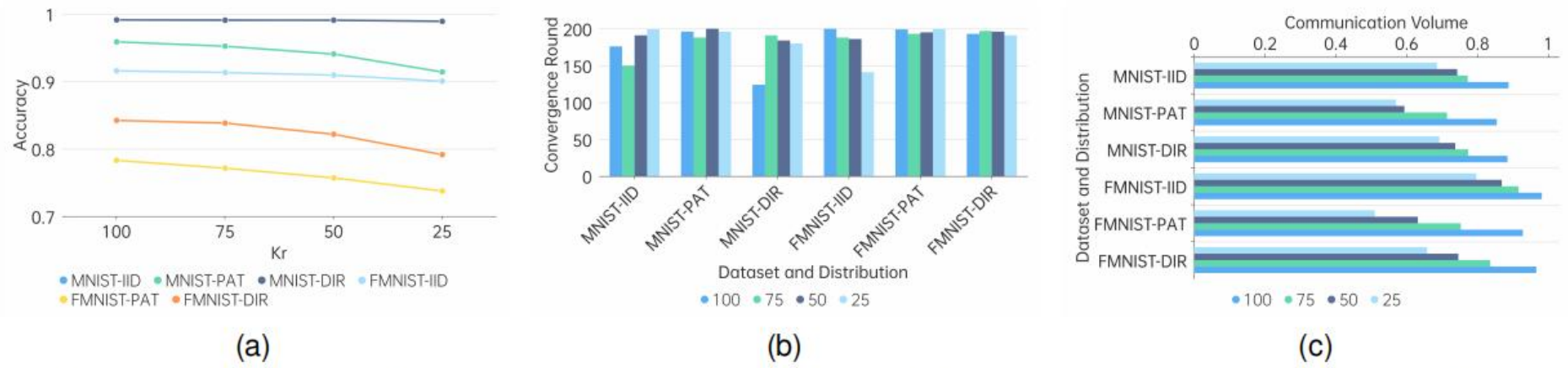


Fig. 5. The metrics when using different K_i^r for different distributions on different datasets. (a) Accuracy; (b) Convergence rate; (c) Communication Volume. We can observe that K_i^r can decrease the communication volume but introduce minor accuracy loss.

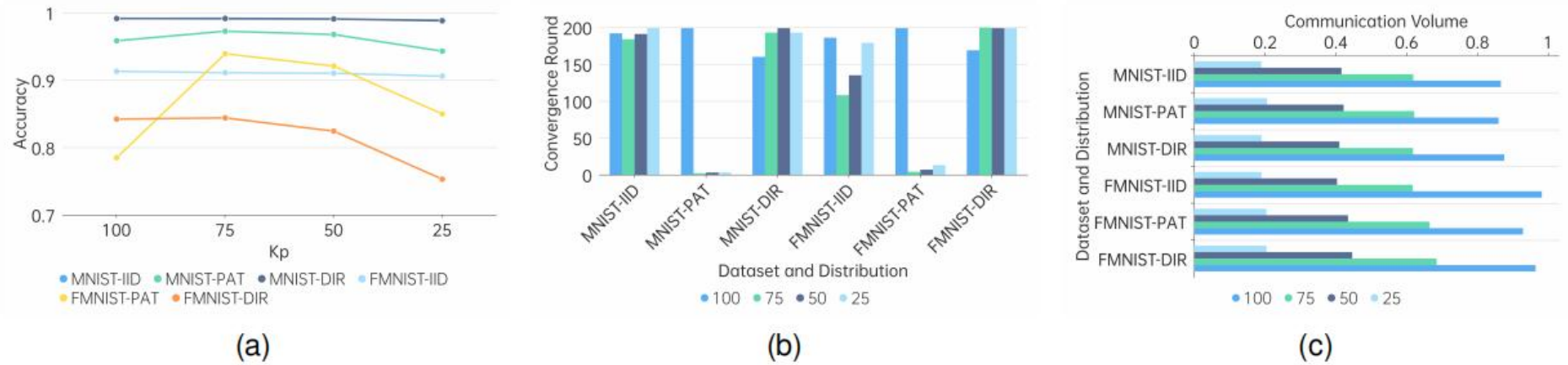
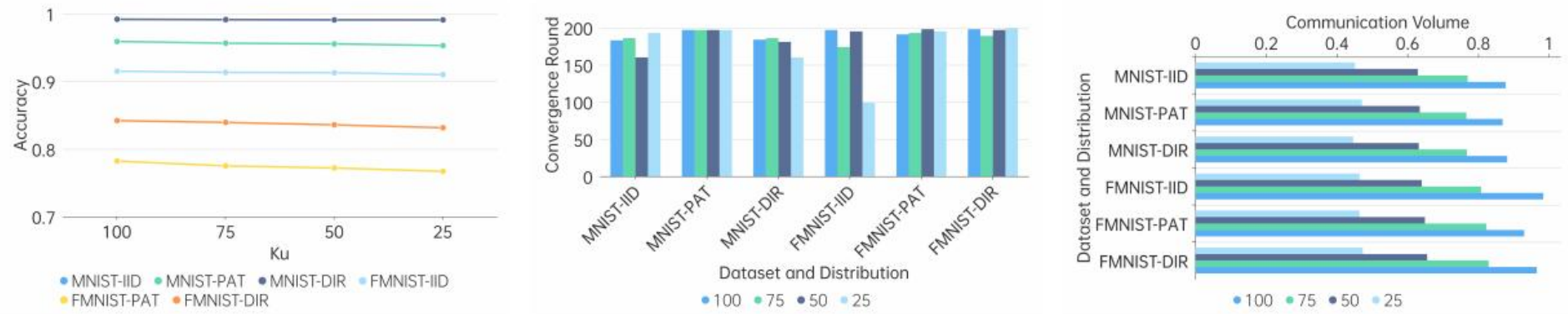


Fig. 6. The metrics when using different K_i^p for different distributions on different datasets. (a) Accuracy; (b) Convergence rate; (c) Communication Volume. We can observe that K_i^p significantly reduces the communication volume but introduce relatively large accuracy loss.

Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

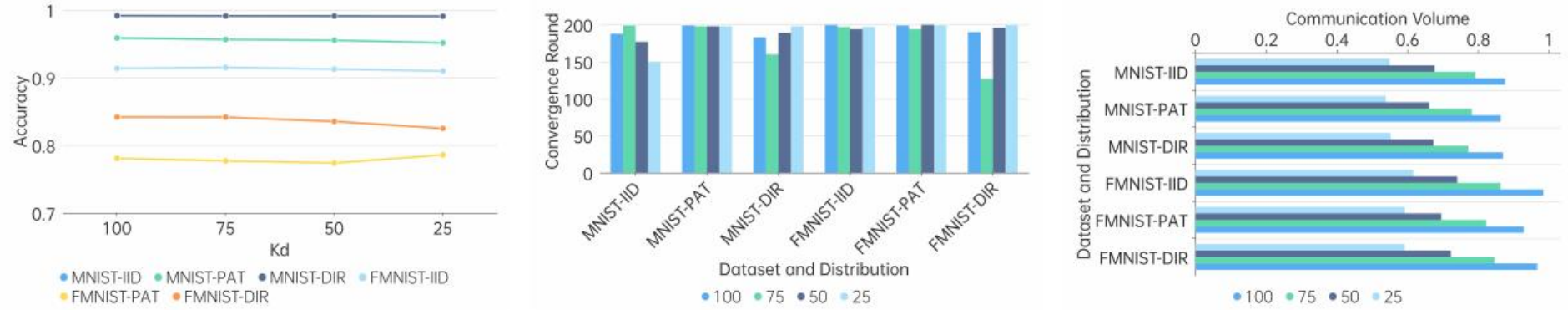
联邦学习效率-模型压缩

- 不同重要性mask下不同压缩率结果
- upload mask和download mask虽然通信量减少有限，但是能够有效维持准确率



(a) (b) (c)

Fig. 7. The metrics when using different K_i^u for different distributions on different datasets. (a) Accuracy; (b) Convergence rate; (c) Communication Volume. We can observe that K_i^u can keep the training accuracy while decreasing the communication volume.



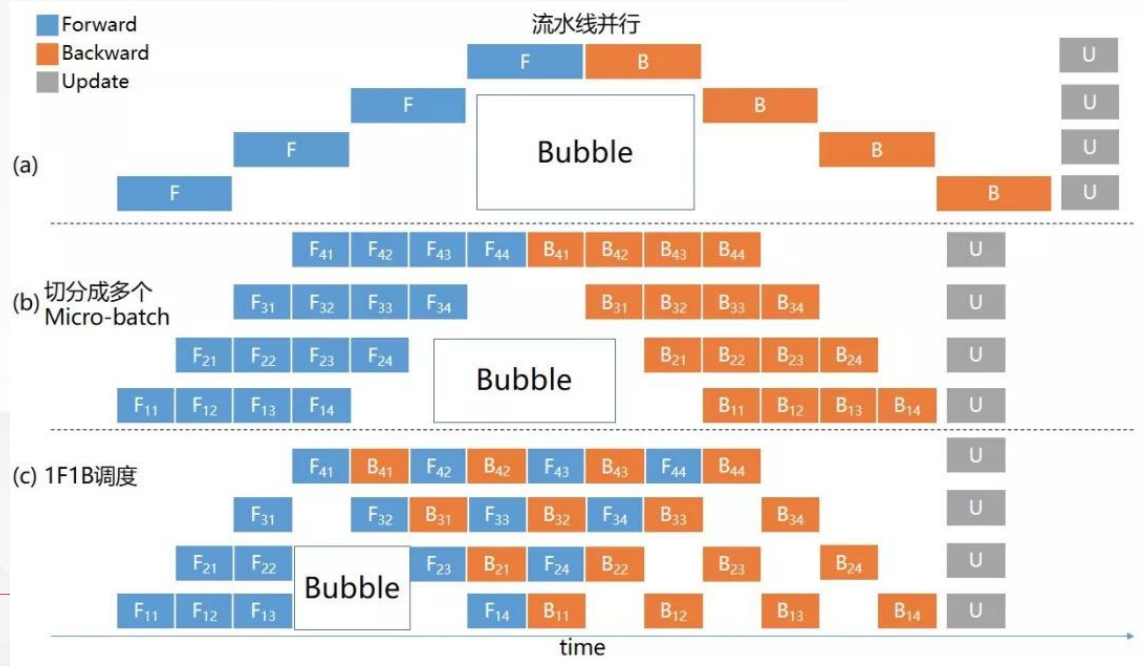
(a) (b) (c)

Fig. 8. The metrics when using different K_i^d for different distributions on different datasets. (a) Accuracy; (b) Convergence rate; (c) Communication Volume. We can observe that K_i^d can keep the training accuracy while decreasing the communication volume.

Hongjian Shi, Jianqing Zhang, Shuming Fan, Ruhui Ma, Haibing Guan, pFedEff: An Efficient and Personalized Federated Cognitive Learning Framework in Multi-agent Systems, IEEE Transactions on Cognitive and Development Systems, 2023.

联邦学习效率-传输调度

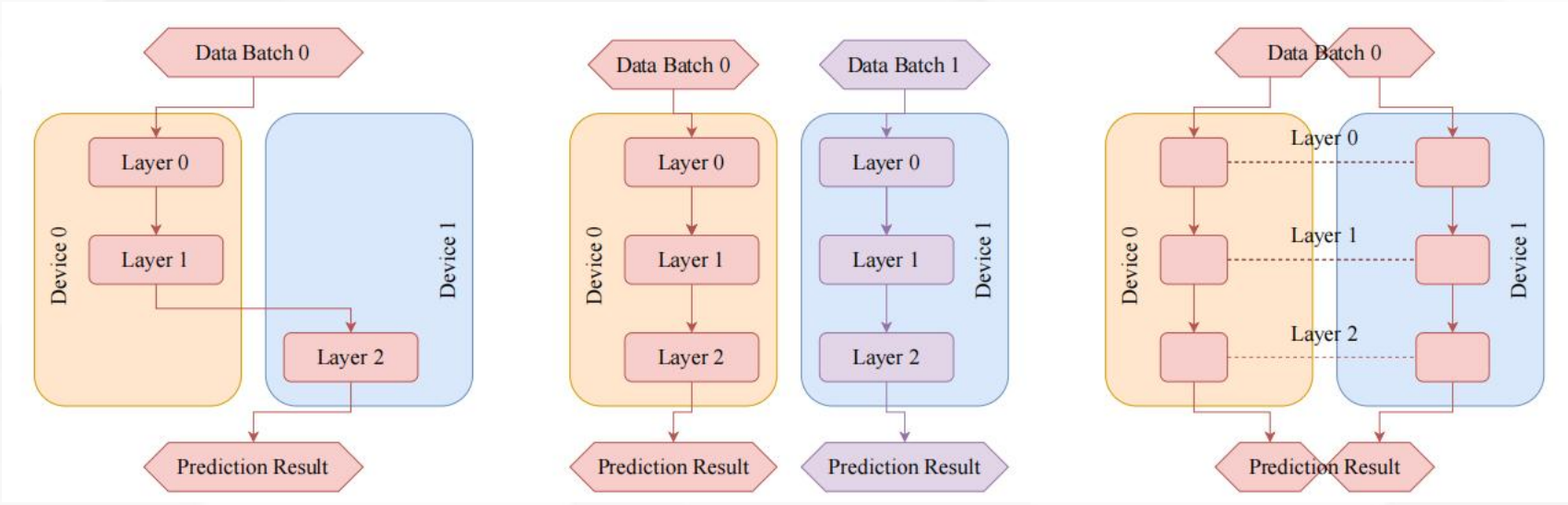
- 任务调度
  - 任务复杂，资源丰富，需要调度计算子任务
- 并行算法：
  - 计算and计算、计算and通信、通信and通信
  - 数据并行、模型并行、算子并行、优化器并行

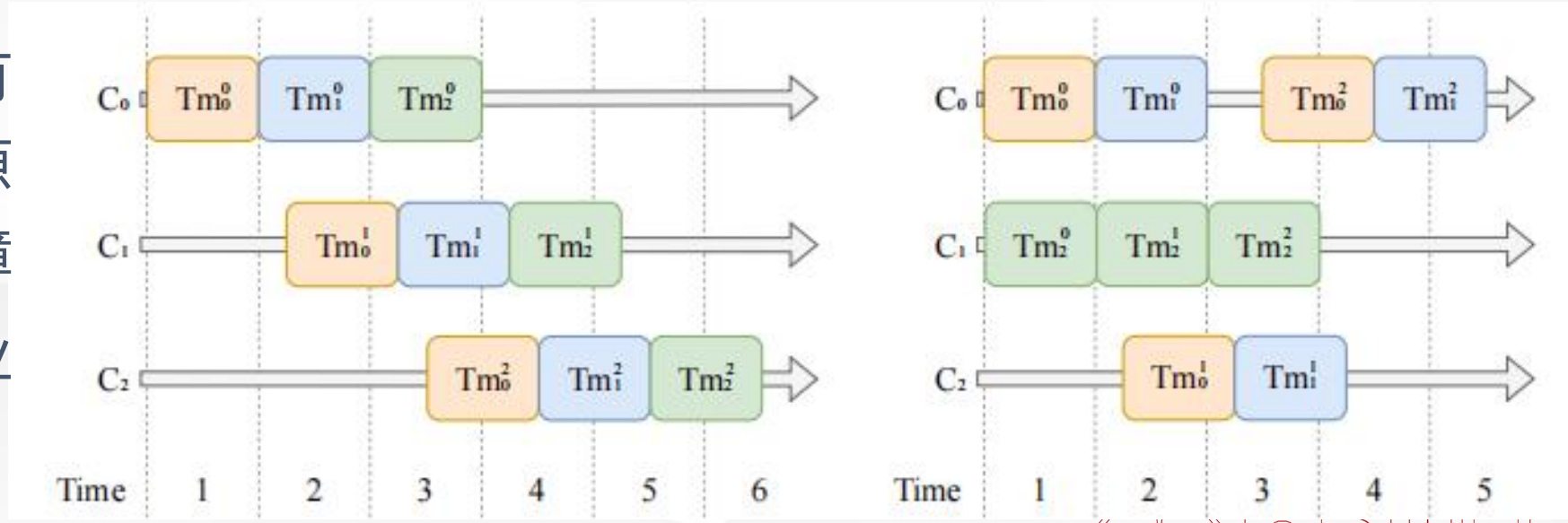


Parallel Strategy	Affiliation
Pangu[2]	Huawei
Pipedream[3]	Microsoft
Gpipe[4]	Google
Fairscale[5]	Facebook
Megatron-LM[6]	NVIDIA
DeepSpeed[7]	Microsoft
Colossal-AI[8]	HPC-AI
FlexFlow[9]	Stanford
OneFlow[10]	OneFlow

联邦学习效率-传输调度

- 并行算法
  - 模型：减少空间占用
  - 数据：减少处理延迟
  - 算子：提高处理速度
- 存在的问题
  - 单一并行算法效率有限，存在大量的资源浪费，可用性无保障
  - 多并行算法需要专业设计

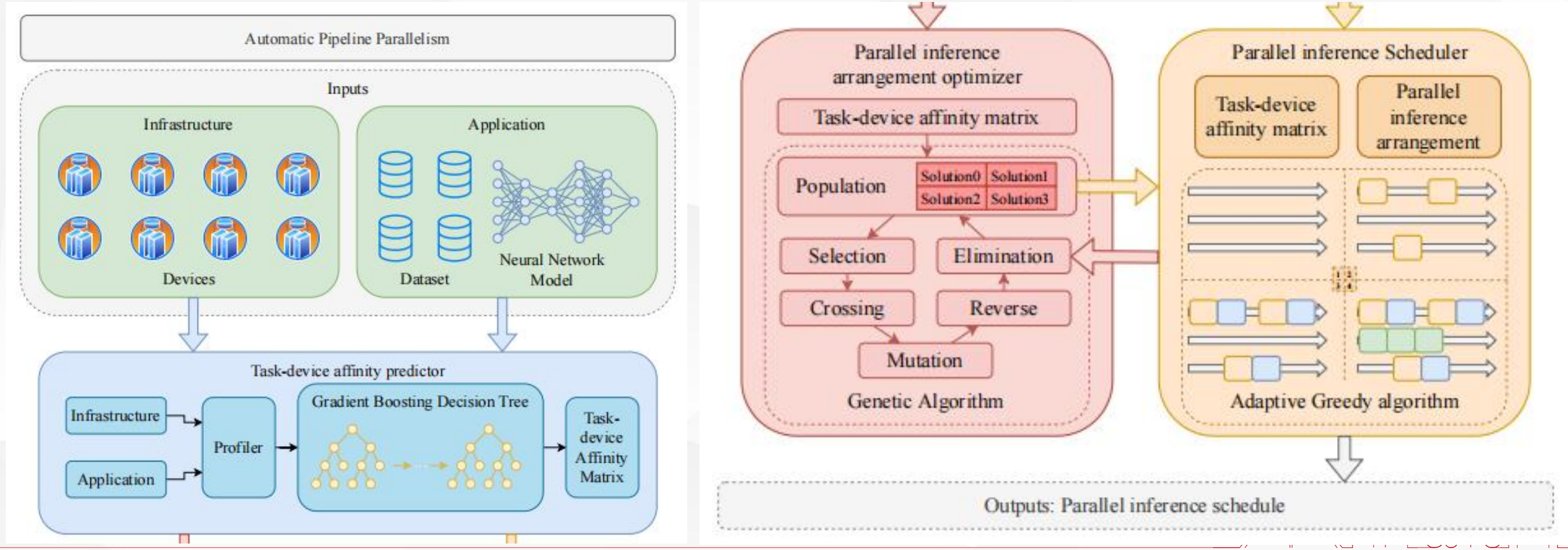




Hongjian Shi, Weichu Zheng, Zifei Liu, Ruhui Ma, Haibing Guan, Automatic Pipeline Parallelism: A Parallel Inference Framework for Deep Learning Applications in 6G Mobile Communication Systems, in IEEE Journal on Selected Areas in Communications，2023.

联邦学习效率-传输调度

- 先采用GBDT预测各任务在各设备上的运行时间，来减少profiling的用时
- 而后采用遗传算法决定任务位置，确认各个任务所在设备，以减少任务处理延迟
- 遗传算法中会用贪心算法排列各设备上的任务，得到适应度函数，并得到实际可运行的任务时间表



Hongjian Shi, Weichu Zheng, Zifei Liu, Ruhui Ma, Haibing Guan, Automatic Pipeline Parallelism: A Parallel Inference Framework for Deep Learning Applications in 6G Mobile Communication Systems, in IEEE Journal on Selected Areas in Communications，2023.

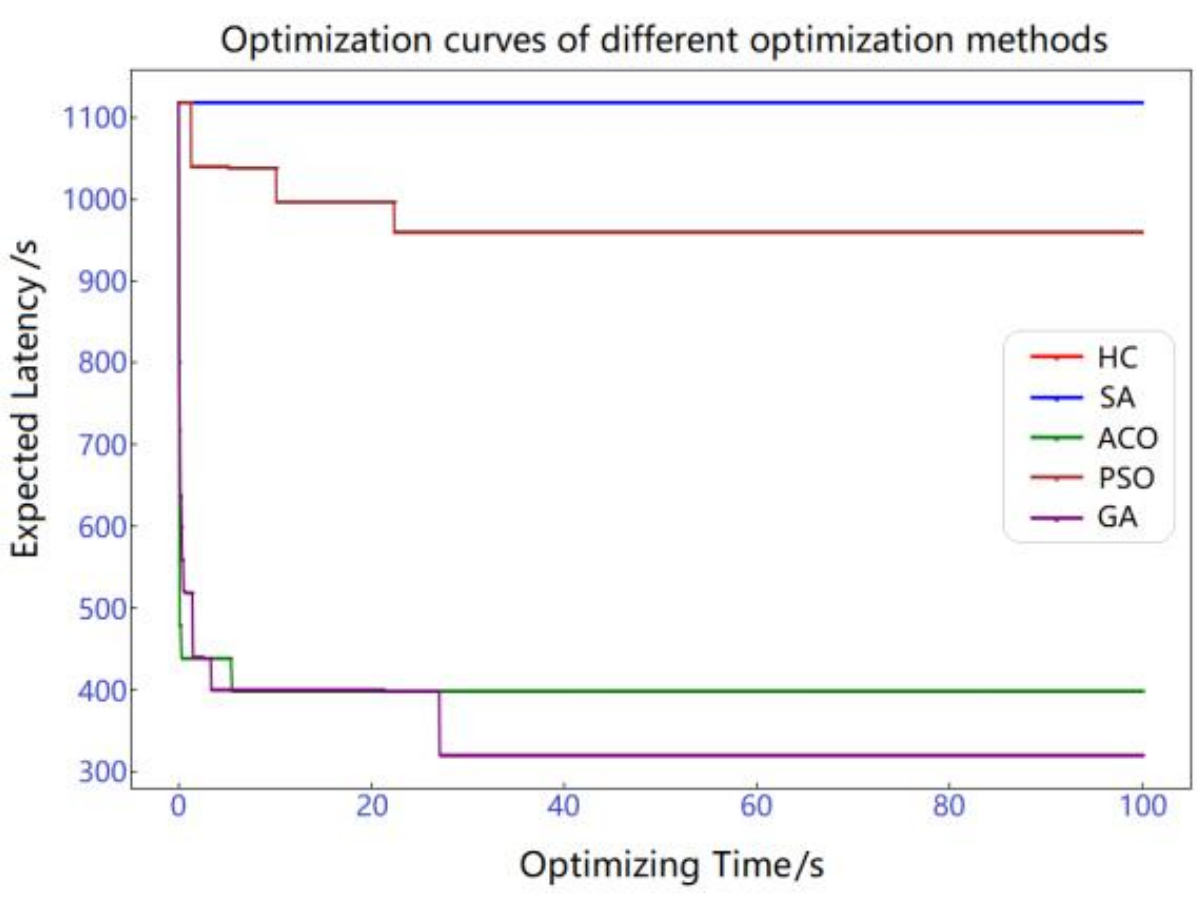
联邦学习效率

- 整体算法能够提高设备利用率
- 减少延迟提高吞吐量
- 同时能提高可用性

- 采用遗传算法的效果和收敛性较好

Method	Latency(s)	Throu.(/s)	Dev. Util.(%)	Rel.(%)
Standalone	6.1053	1310.34	12.2831	0
Model#8	9.6717	827.15	8.3009	0
Data	0.9468	8449.12	83.9462	100
Pipeline#8	2.9603	2702.43	27.0113	0
AP^2#5	1.3739	5822.89	49.9417	100

Method	Latency(s)	Throu.(/s)	Dev. Util.(%)	Rel.(%)
Standalone	6.9814	1432.37	9.8450	0
Model#5	11.9720	835.29	6.2150	0
Data	0.9734	10273.53	82.3580	100
Pipeline#5	1.7692	5652.16	39.2073	89
AP^2#5	1.7178	5821.57	45.4050	100



Hongjian Shi, Weichu Zheng, Zifei Liu, Ruhui Ma, Haibing Guan, Automatic Pipeline Parallelism: A Parallel Inference Framework for Deep Learning Applications in 6G Mobile Communication Systems, in IEEE Journal on Selected Areas in Communications，2023.

联邦学习效率

- GBDT预测方法准确率明显由于其他预测方法
- 并且由于是线下训练，训练时间不会对线上推理产生负面影响
- 主要是由于GBDT对于表格数据和feature较少的数据有着十分优越的表现

Method	Test Accuracy	Train Accuracy	Training time(ms)
GBDT	**0.7176±0.0001**	**0.8058±0.0000**	107.48±0.64
FCNN-leakyrelu	0.6583±0.0003	0.7670±0.0000	6620.46±108.63
FCNN-relu [78]	0.6719±0.0002	0.7664±0.0000	6699.71±114.25
FCNN-tanh	0.6860±0.0000	0.7674±0.0000	6674.88±95.96
FCNN-sigmoid	0.7039±0.0000	0.7624±0.0000	6574.73±106.43
ABDT [79]	0.7114±0.0000	0.8001±0.0000	70.20±1.06
RF [80]	0.7096±0.0011	0.7738±0.0004	9.56±0.57
SVR [81]	0.7166±0.0000	0.7399±0.0000	**7.97±0.52**

n_estimators	Train Accuracy	Test Accuracy	Training Time(ms)
100	0.7410±0.0000	0.7172±0.0000	**48.39±0.72**
120	0.7634±0.0000	0.7251±0.0000	58.17±0.57
140	0.7783±0.0000	**0.7266±0.0000**	67.46±0.79
160	0.7882±0.0000	0.7238±0.0000	76.44±0.68
180	0.7978±0.0000	0.7209±0.0000	86.48±1.20
200	0.8058±0.0000	0.7176±0.0001	95.27±0.77
220	0.8120±0.0000	0.7132±0.0001	106.06±4.01
240	0.8219±0.0025	0.7064±0.0029	115.31±3.25
260	0.8273±0.0020	0.7020±0.0025	125.07±4.59
280	**0.8309±0.0000**	0.6992±0.0036	132.34±1.07

Hongjian Shi, Weichu Zheng, Zifei Liu, Ruhui Ma, Haibing Guan, Automatic Pipeline Parallelism: A Parallel Inference Framework for Deep Learning Applications in 6G Mobile Communication Systems, in IEEE Journal on Selected Areas in Communications，2023.

联邦学习安全

- 拜占庭攻击 - 双端协同预警
- 不可信第三方 - 区块链辅助聚合

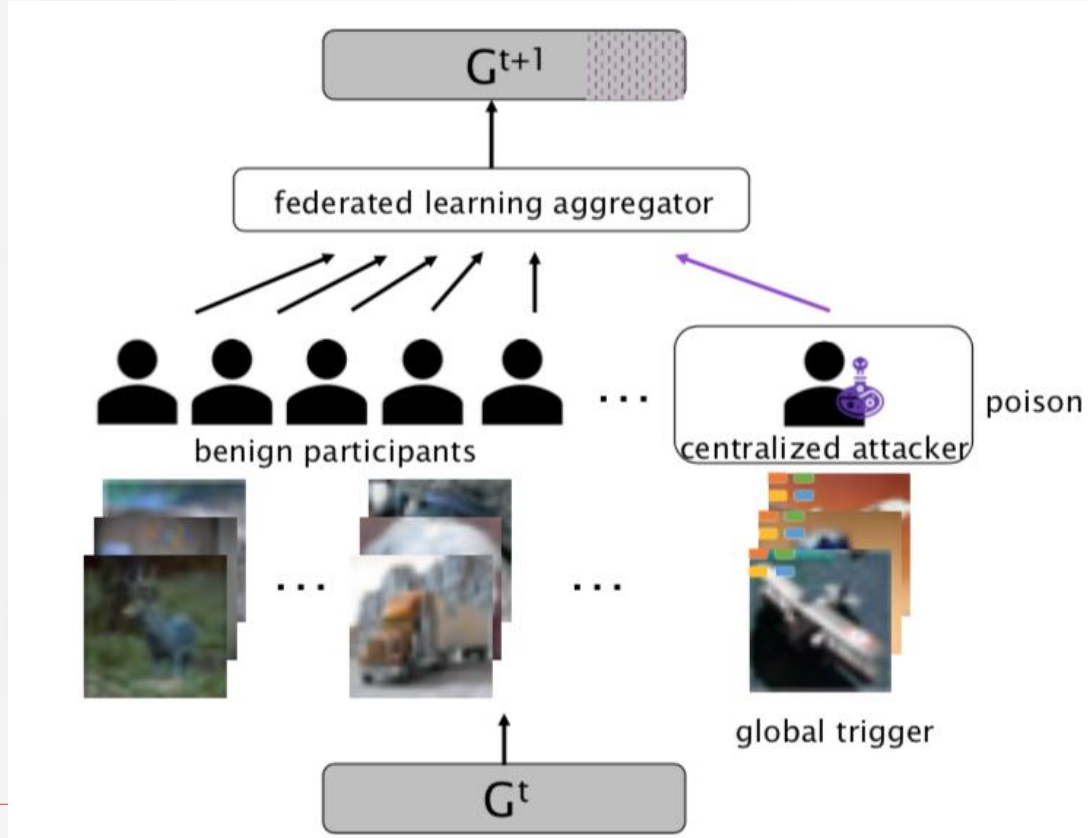
联邦学习安全---拜占庭攻击

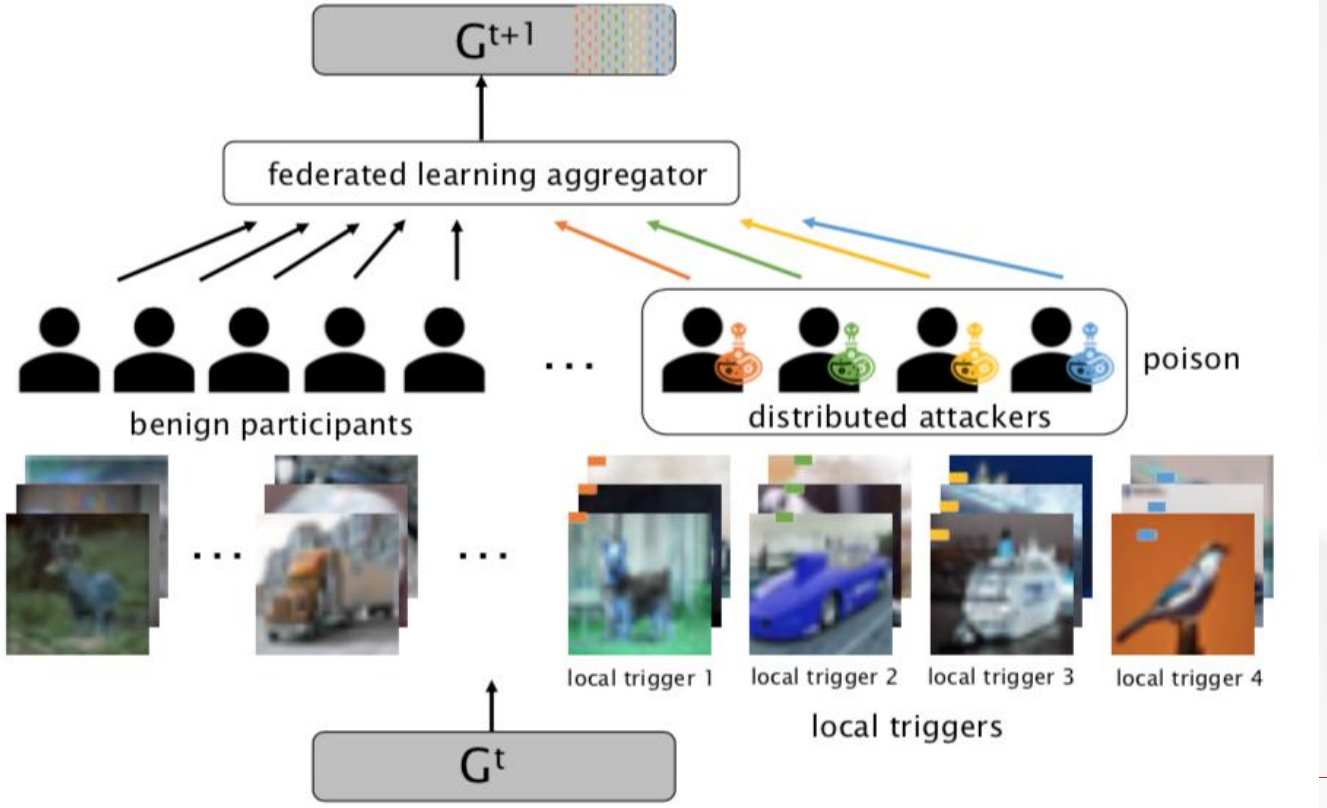
计算安全

- 拜占庭攻击，系统中存在恶意节点通过传输错误数据破坏训练模型

异常更新判断：

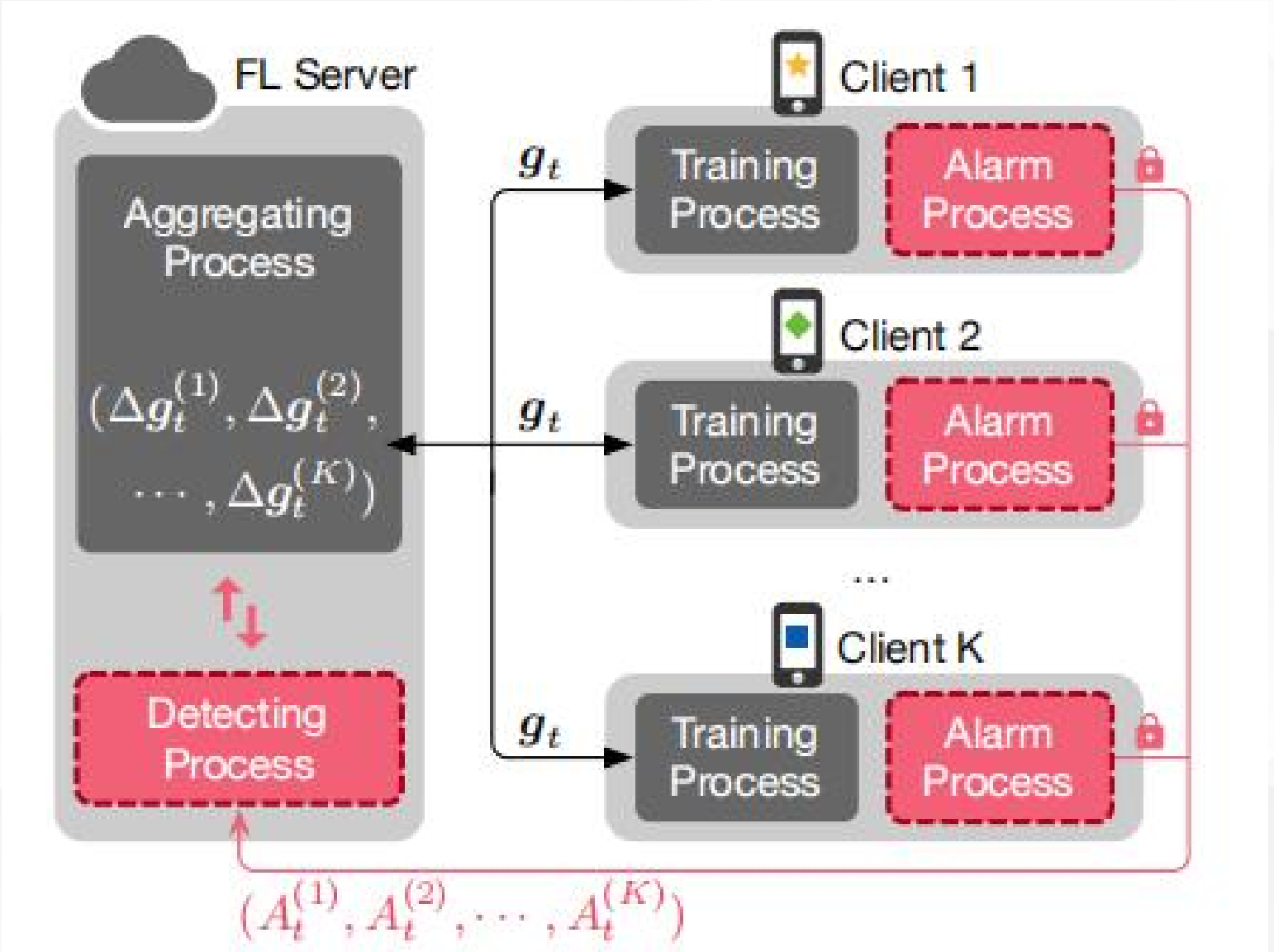
- 判断模型更新中的离群点
- 剔除离群点或是抑制离群点贡献





联邦学习安全---拜占庭攻击

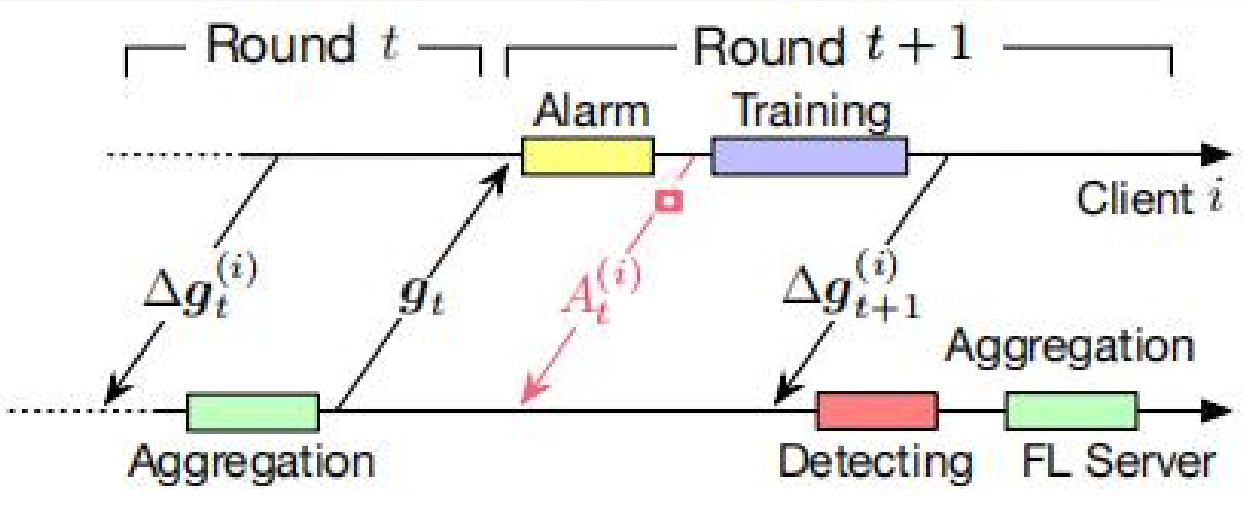
- 利用客户端信息辅助排除恶意节点
- 客户端侧拥有**示警机制**，可以和服务器侧的决策流程互相配合
- 服务器侧使用基于准确度筛查和权重分析的**决策流程**
- 系统提供两种辅助机制，分别为**惩罚机制**和**奖励机制**

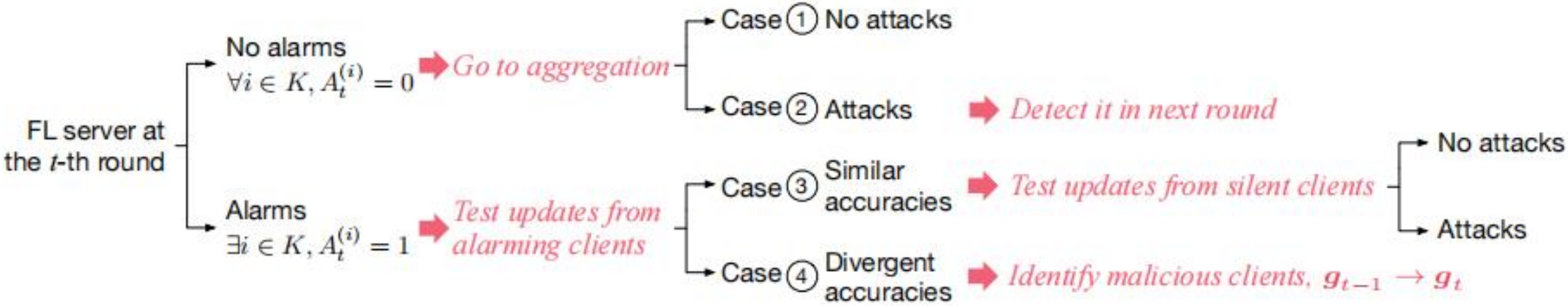


Hanxi Guo, Hao Wang, Tao Song, Yang Hua, Zhangcheng Lv, Xiulang Jin, Zhengui Xue, Ruhui Ma, Haibing Guan, Siren: Byzantine-robust Federated Learning via Proactive Alarming, Proceedings of ACM Symposium on Cloud Computing, pp. 47--60, 2021.

联邦学习安全---拜占庭攻击

- 客户端侧结构与方法流程
  - 根据全局模型和本地模型**区别**进行判断
- 服务器侧结构与方法流程
  - 根据客户端**告警**判断是否存在恶意攻击





Hanxi Guo, Hao Wang, Tao Song, Yang Hua, Zhangcheng Lv, Xiulang Jin, Zhengui Xue, Ruhui Ma, Haibing Guan, Siren: Byzantine-robust Federated Learning via Proactive Alarming, Proceedings of ACM Symposium on Cloud Computing, pp. 47--60, 2021.

联邦学习安全---拜占庭攻击

- 在sign-flipping, label-flipping和targeted model poisoning攻击下可以有效收敛到一个较优准确率
- 也可以看出针对targeted model poisoning时，Siren稳定性较好

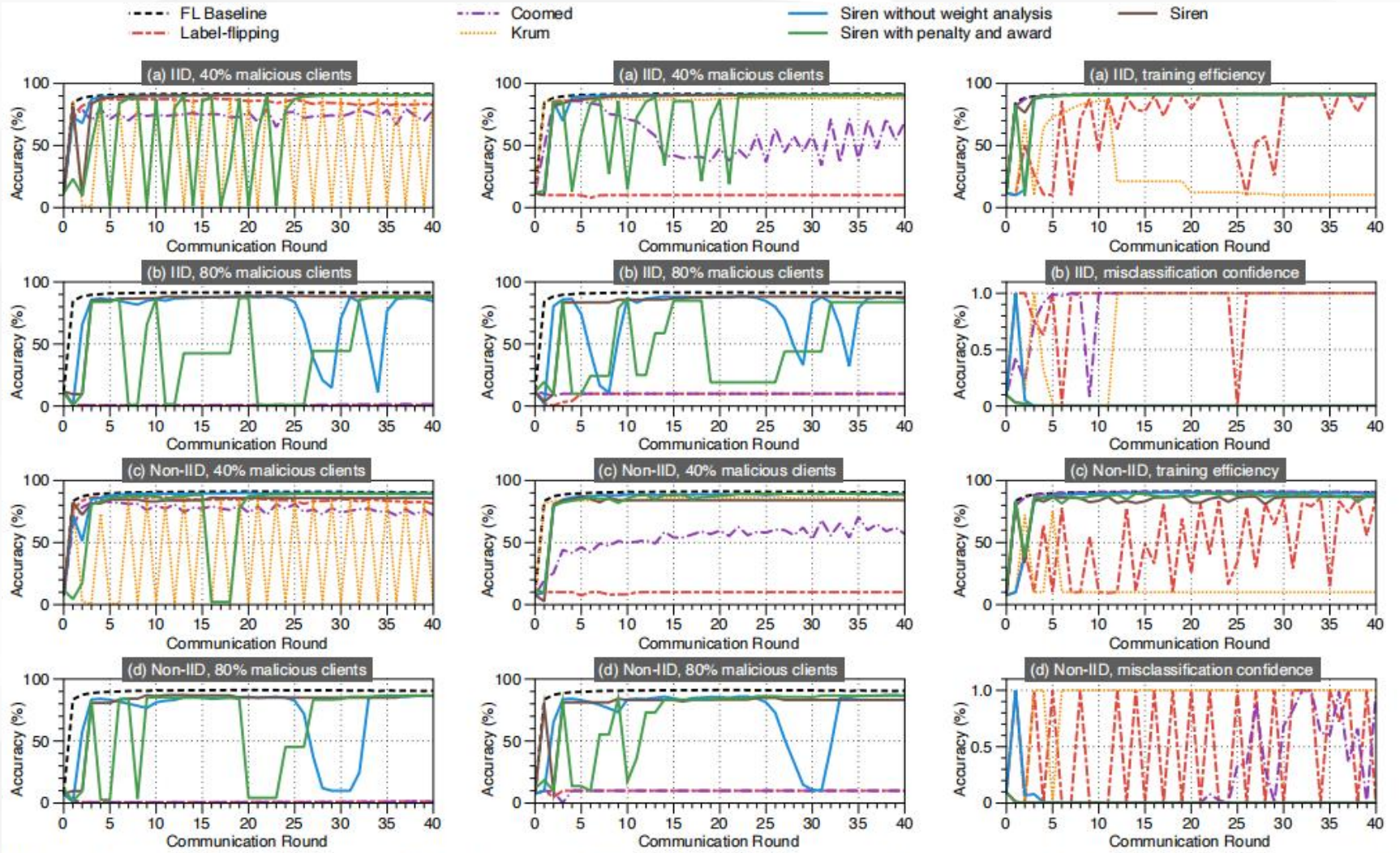


Figure 5: Training efficiency under label-flipping attack when $|K| = 10$.

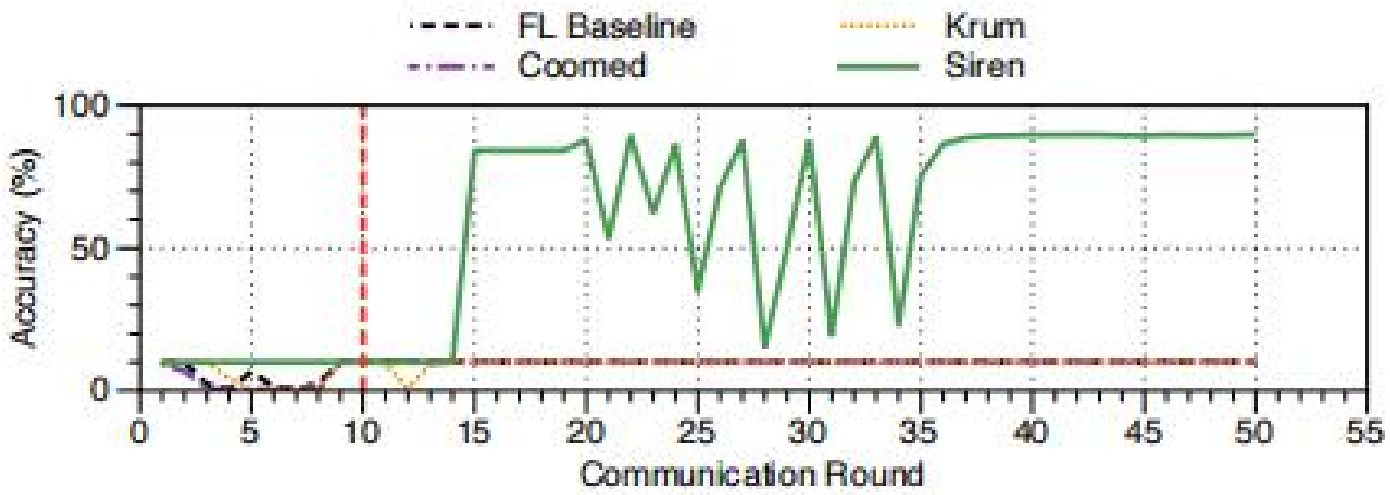
Figure 6: Training efficiency under sign-flipping attack when $|K| = 10$.

Figure 7: Training efficiency and misclassification confidence under targeted model poisoning, $|K| = 10$.

Hanxi Guo, Hao Wang, Tao Song, Yang Hua, Zhangcheng Lv, Xiulang Jin, Zhengui Xue, Ruhui Ma, Haibing Guan, Siren: Byzantine-robust Federated Learning via Proactive Alarming, Proceedings of ACM Symposium on Cloud Computing, pp. 47--60, 2021.

联邦学习安全---拜占庭攻击

- Siren可以在大比例恶意客户端的情况下（80%）拥有较强的防御能力
- Siren可以在系统已经遭受攻击的情况下恢复原有系统的训练准确率



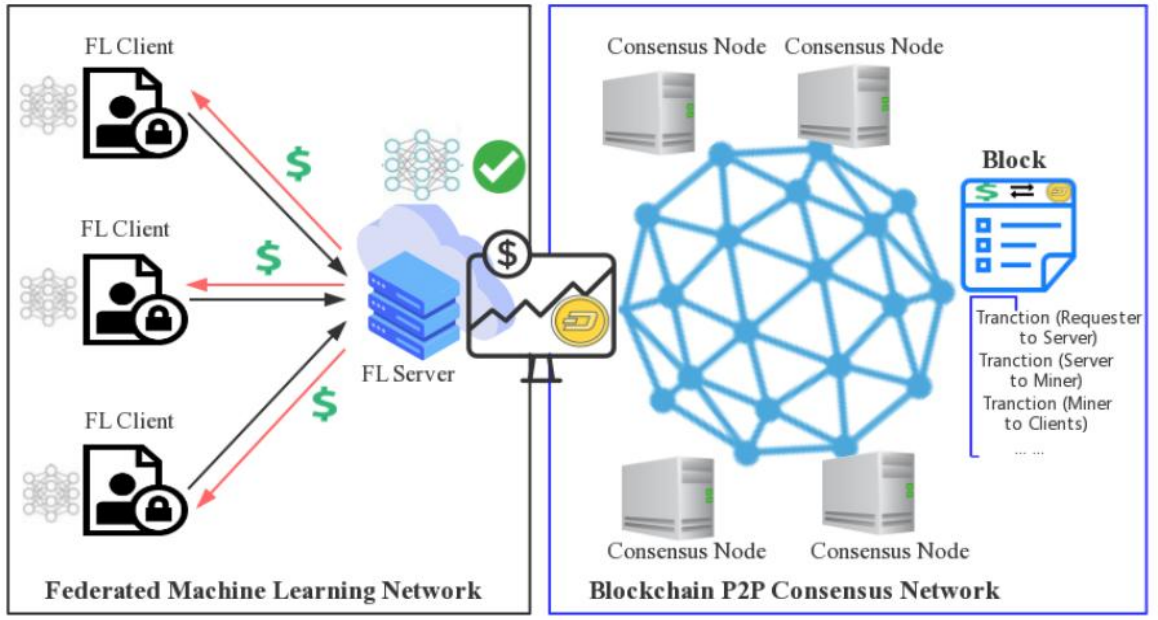
Attack Type	Defense Methods	Malicious Proportion	Accuracy
None	None	0%	55.33%
Sign-flipping	None	40%	10.01%
		80%	10.01%
	Krum[1]	40%	41.53%
	Coomed	40%	34.25%
		80%	9.99%
	SIREN	40%	**51.58%**
		80%	**45.50%**
Label-flipping	None	40%	34.70%
		80%	11.68%
	Krum[1]	40%	44.72%
	Coomed	40%	37.21%
		80%	9.60%
	SIREN	40%	**49.82%**
		80%	**43.52%**

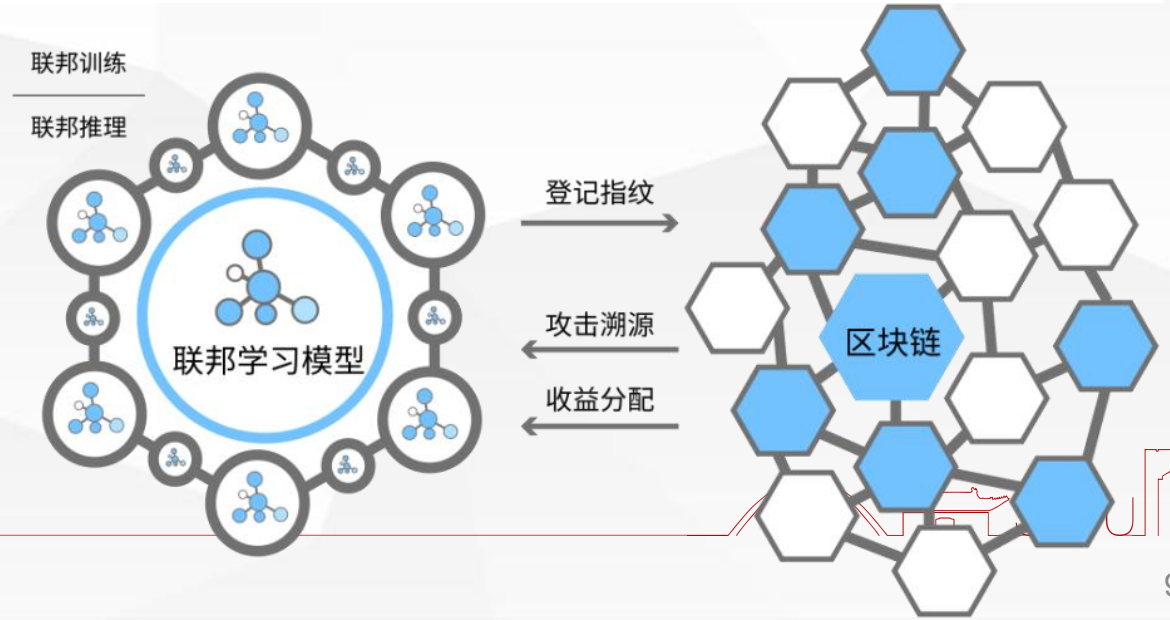
[1] Krum cannot work properly when malicious clients' proportion reaches 80%.

Hanxi Guo, Hao Wang, Tao Song, Yang Hua, Zhangcheng Lv, Xiulang Jin, Zhengui Xue, Ruhui Ma, Haibing Guan, Siren: Byzantine-robust Federated Learning via Proactive Alarming, Proceedings of ACM Symposium on Cloud Computing, pp. 47--60, 2021.

联邦学习安全---不可信第三方

- 通信安全
  - 无可信第三方服务器，易受攻击
- 联邦学习+区块链
  - 用区块链替换原有server保存全局模型，可以提高全局模型的防篡改能力
  - 区块链本身的共识机制和联邦学习的聚合权重有很多相似之处



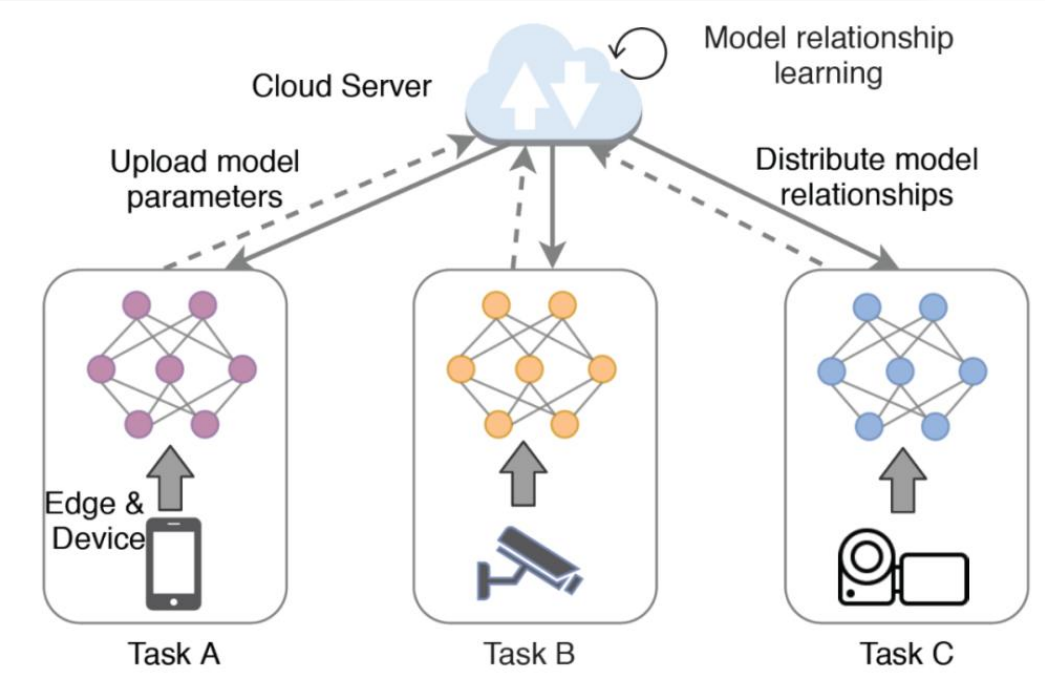


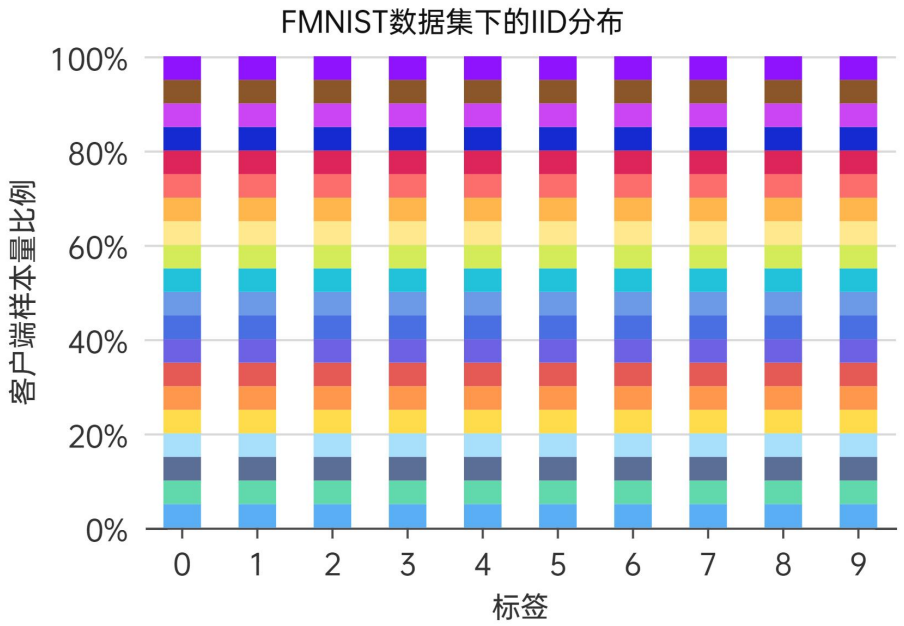
联邦学习异构性

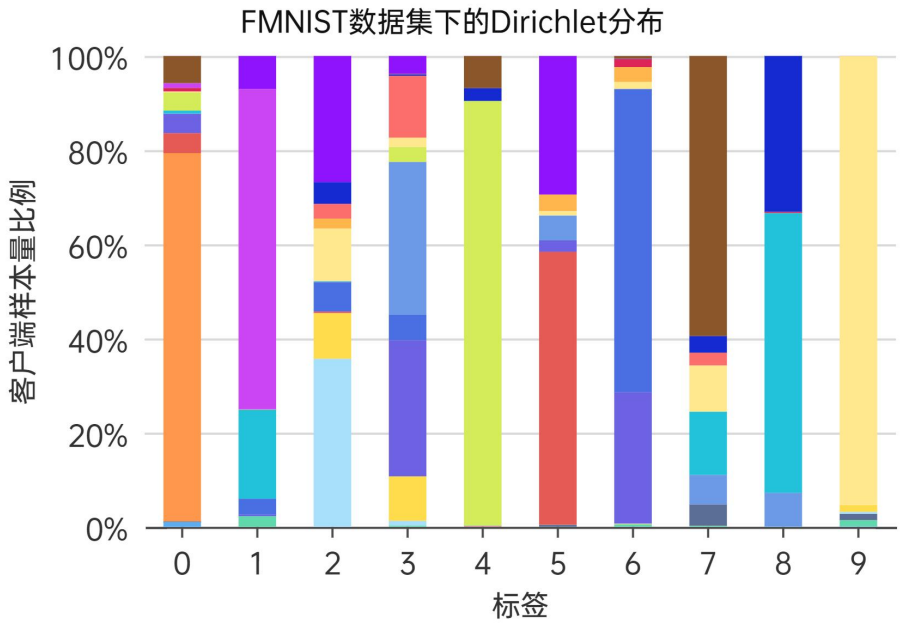
- 统计异构性 - 非聚合联邦学习
- 系统异构性 - 适应性训练调整

联邦学习异构性---统计异构性

- 统计异构
  - 各客户端数据分布不同，模型不准确
- 跨域联邦学习：
  - 模型之间交换知识，不单纯进行聚合
  - 模型在本地测试集上的效果会更好







联邦学习异构性---统计异构性

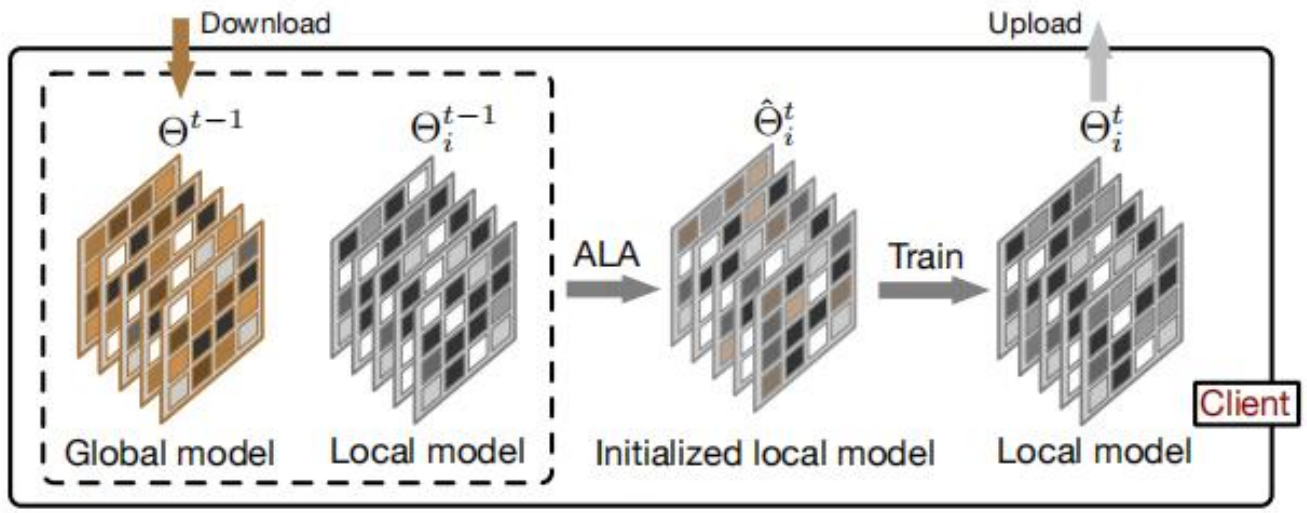
- 引入个性化mask做element-wise的更新

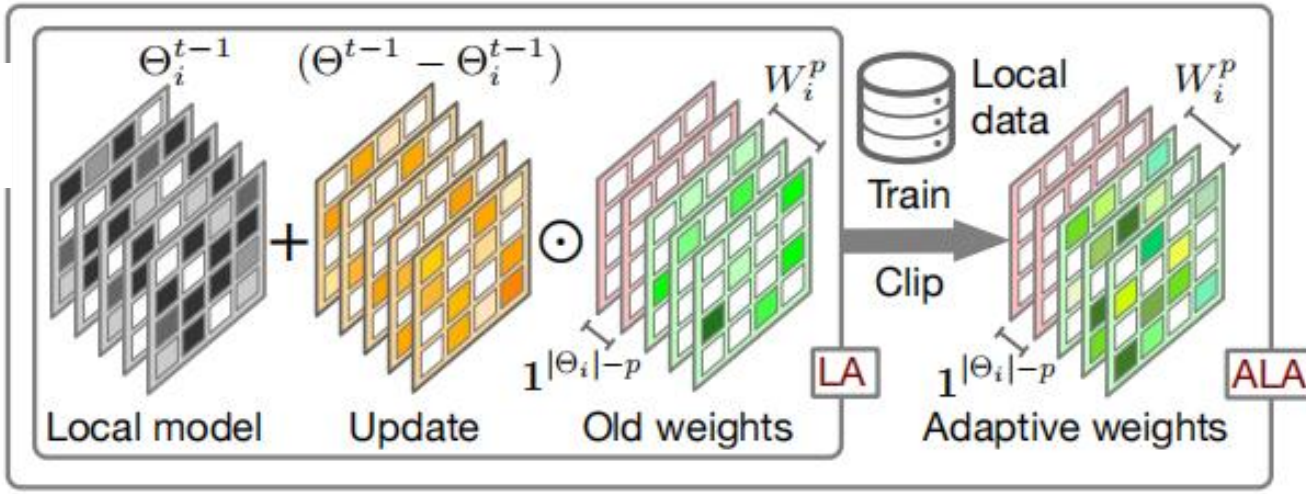
$$\hat{\Theta}_i^t := \Theta_i^{t-1} \odot W_{i,1} + \Theta^{t-1} \odot W_{i,2},$$
$$s.t. \quad w_1^q + w_2^q = 1, \forall \text{ valid } q$$

$$\hat{\Theta}_i^t := \Theta_i^{t-1} + (\Theta^{t-1} - \Theta_i^{t-1}) \odot W_i,$$

$$\hat{\Theta}_i^t := \Theta_i^{t-1} + (\Theta^{t-1} - \Theta_i^{t-1}) \odot [\mathbf{1}^{|\Theta_i|-p}; W_i^p],$$

$$W_i^p \leftarrow W_i^p - \eta \nabla_{W_i^p} \mathcal{L}(\hat{\Theta}_i^t, D_i^{s,t}; \Theta^{t-1}),$$





Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Haibing Guan, FedALA: Adaptive Local Aggregation for Personalized Federated Learning, Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2022.

联邦学习异构性---统计异构性

- 在多个数据集多种设定上均取得了最高的准确率

Table 2: The test accuracy (%) in the pathological heterogeneous setting and practical heterogeneous setting.

Settings	Pathological heterogeneous setting			Practical heterogeneous setting				
Methods	MNIST	Cifar10	Cifar100	Cifar10	Cifar100	TINY	TINY*	AG News
FedAvg	97.93±0.05	55.09±0.83	25.98±0.13	59.16±0.47	31.89±0.47	19.46±0.20	19.45±0.13	79.57±0.17
FedProx	98.01±0.09	55.06±0.75	25.94±0.16	59.21±0.40	31.99±0.41	19.37±0.22	19.27±0.23	79.35±0.23
FedAvg-C	99.79±0.00	92.13±0.03	66.17±0.03	90.34±0.01	51.80±0.02	30.67±0.08	36.94±0.10	95.89±0.25
FedProx-C	99.80±0.04	92.12±0.03	66.07±0.08	90.33±0.01	51.84±0.07	30.77±0.13	38.78±0.52	96.10±0.22
Per-FedAvg	99.63±0.02	89.63±0.23	56.80±0.26	87.74±0.19	44.28±0.33	25.07±0.07	21.81±0.54	93.27±0.25
FedRep	99.77±0.03	91.93±0.14	67.56±0.31	90.40±0.24	52.39±0.35	37.27±0.20	39.95±0.61	96.28±0.14
pFedMe	99.75±0.02	90.11±0.10	58.20±0.14	88.09±0.32	47.34±0.46	26.93±0.19	33.44±0.33	91.41±0.22
Ditto	99.81±0.00	92.39±0.06	67.23±0.07	90.59±0.01	52.87±0.64	32.15±0.04	35.92±0.43	95.45±0.17
FedAMP	99.76±0.02	90.79±0.16	64.34±0.37	88.70±0.18	47.69±0.49	27.99±0.11	29.11±0.15	94.18±0.09
FedPHP	99.73±0.00	90.01±0.00	63.09±0.04	88.92±0.02	50.52±0.16	35.69±3.26	29.90±0.51	94.38±0.12
FedFomo	99.83±0.00	91.85±0.02	62.49±0.22	88.06±0.02	45.39±0.45	26.33±0.22	26.84±0.11	95.84±0.15
APPLE	99.75±0.01	90.97±0.05	65.80±0.08	89.37±0.11	53.22±0.20	35.04±0.47	39.93±0.52	95.63±0.21
PartialFed	99.86±0.01	89.60±0.13	61.39±0.12	87.38±0.08	48.81±0.20	35.26±0.18	37.50±0.16	85.20±0.16
FedALA	**99.88±0.01**	**92.44±0.02**	**67.83±0.06**	**90.67±0.03**	**55.92±0.03**	**40.54±0.02**	**41.94±0.05**	**96.52±0.08**

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Haibing Guan, FedALA: Adaptive Local Aggregation for Personalized Federated Learning, Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2022.

联邦学习异构性---统计异构性

FedALA对在不同统计异构性和扩展性的情况都展开了测试，在多种设定下其效果均比传统方法要好。

	Heterogeneity			Scalability		Applicability of ALA			
Datasets	Tiny-ImageNet		AG News	Cifar100		Tiny-ImageNet		Cifar100	
Methods	$Dir(0.01)$	$Dir(0.5)$	$Dir(1)$	50 clients	100 clients	Acc.	Imps.	Acc.	Imps.
FedAvg	15.70±0.46	21.14±0.47	87.12±0.19	31.90±0.27	31.95±0.37	40.54±0.17	21.08	55.92±0.15	24.03
FedProx	15.66±0.36	21.22±0.47	87.21±0.13	31.94±0.30	31.97±0.24	40.53±0.26	21.16	56.18±0.65	24.19
FedAvg-C	49.88±0.11	16.21±0.05	91.38±0.21	49.82±0.11	47.90±0.12	—	—	—	—
FedProx-C	49.84±0.02	16.36±0.19	92.03±0.19	49.79±0.14	48.02±0.02	—	—	—	—
Per-FedAvg	39.39±0.30	16.36±0.13	87.08±0.26	44.31±0.20	36.07±0.24	30.90±0.28	5.83	48.68±0.36	4.40
FedRep	55.43±0.15	16.74±0.09	92.25±0.20	47.41±0.18	44.61±0.20	37.89±0.31	0.62	53.02±0.11	0.63
pFedMe	41.45±0.14	17.48±0.61	87.08±0.18	48.36±0.64	46.45±0.18	27.30±0.24	0.37	47.91±0.21	0.57
Ditto	50.62±0.02	18.98±0.05	91.89±0.17	54.22±0.04	52.89±0.22	40.75±0.06	8.60	56.33±0.07	3.46
FedAMP	48.42±0.06	12.48±0.21	83.35±0.05	44.39±0.35	40.43±0.17	28.18±0.20	0.19	48.03±0.23	0.34
FedPHP	48.63±0.02	21.09±0.07	90.52±0.19	52.44±0.16	49.70±0.31	40.16±0.24	4.47	54.28±0.21	3.76
FedFomo	46.36±0.54	11.59±0.11	91.20±0.18	42.56±0.33	38.91±0.08	—	—	—	—
APPLE	48.04±0.10	24.28±0.21	84.10±0.18	55.06±0.20	52.81±0.29	—	—	—	—
PartialFed	49.38±0.02	24.20±0.10	91.01±0.28	48.95±0.07	39.31±0.01	35.40±0.02	0.14	48.99±0.05	0.18
FedALA	**55.75±0.02**	**27.85±0.06**	**92.45±0.10**	**55.61±0.02**	**54.68±0.57**	—	—	—	—

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Haibing Guan, FedALA: Adaptive Local Aggregation for Personalized Federated Learning, Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2022.

联邦学习异构性---统计异构性

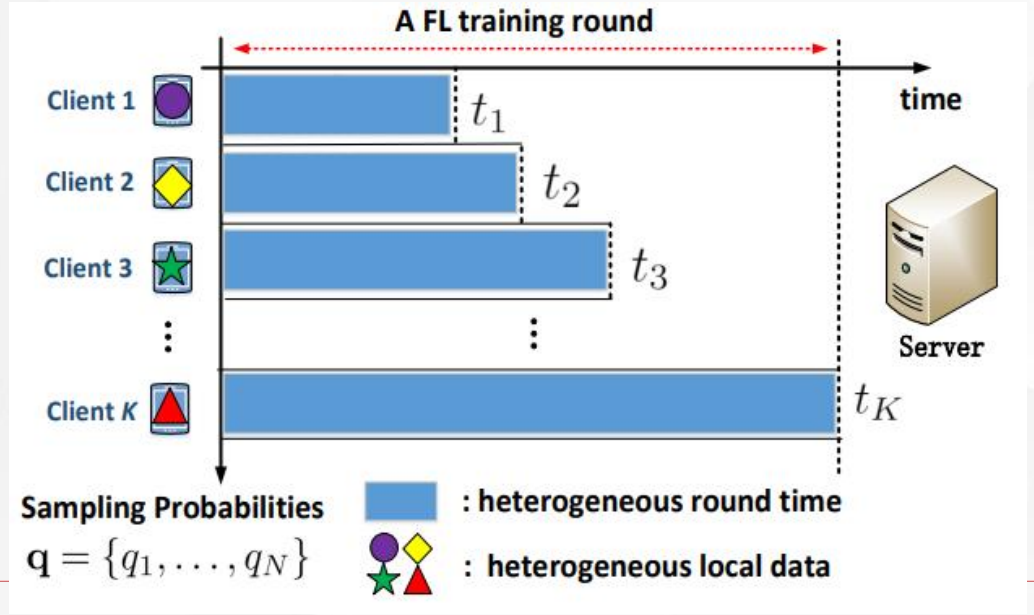
- 同时算法相比其他baseline，减少了一定的时间，提高了整体的计算效率和训练效率
- 其通信量和FedAvg相当，并没有引入额外的传输

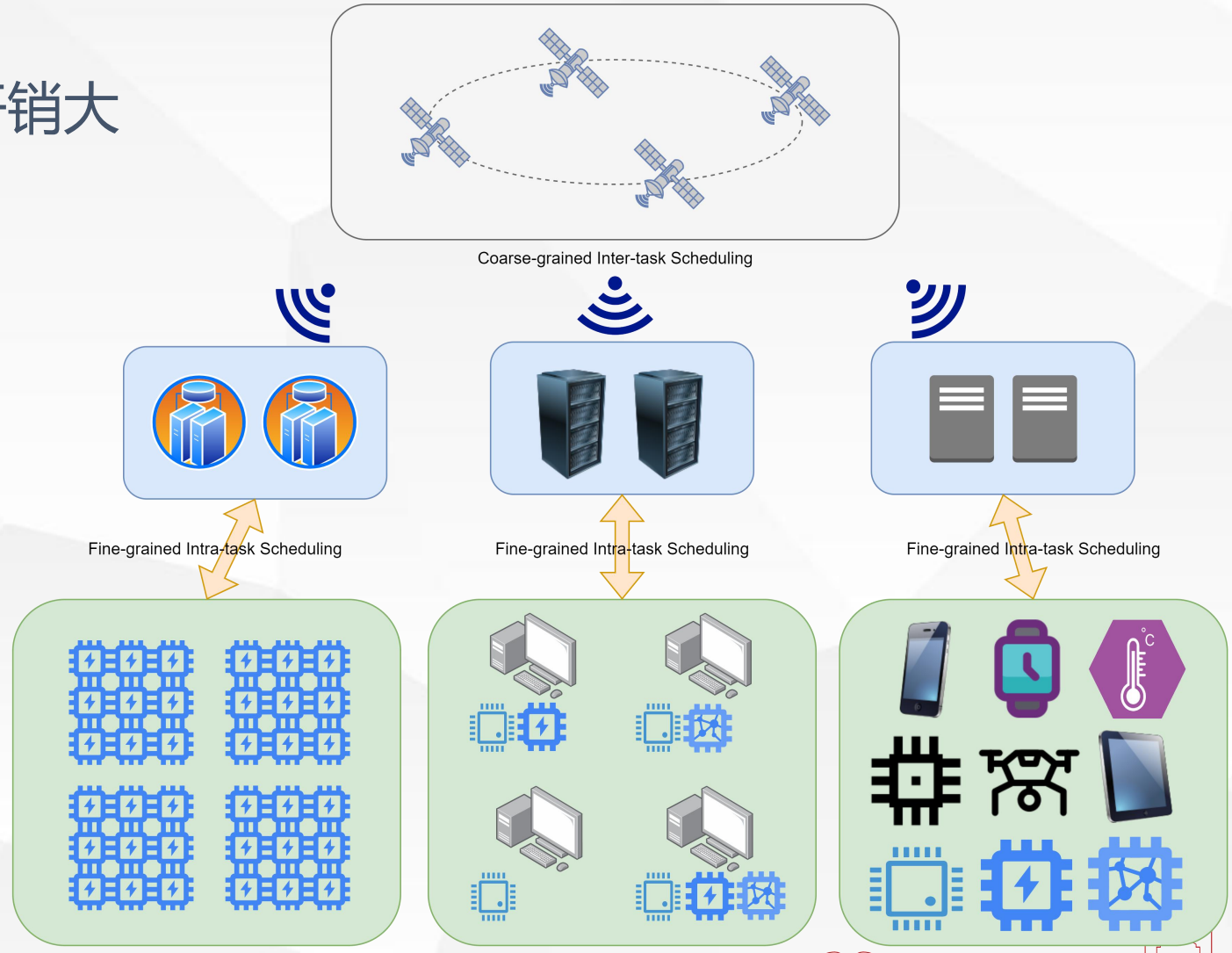
	Computation		Communication
	Total time	Time/iter.	Param./iter.
FedAvg	365 min	1.59 min	$2 * \Sigma$
FedProx	325 min	1.99 min	$2 * \Sigma$
FedAvg-C	607 min	24.28 min	$2 * \Sigma$
FedProx-C	711 min	28.44 min	$2 * \Sigma$
Per-FedAvg	121 min	3.56 min	$2 * \Sigma$
FedRep	471 min	4.09 min	$2 * \alpha_f * \Sigma$
pFedMe	1157 min	10.24 min	$2 * \Sigma$
Ditto	318 min	11.78 min	$2 * \Sigma$
FedAMP	92 min	1.53 min	$2 * \Sigma$
FedPHP	264 min	4.06 min	$2 * \Sigma$
FedFomo	193 min	2.72 min	$(1 + M) * \Sigma$
APPLE	132 min	2.93 min	$(1 + M) * \Sigma$
PartialFed	693 min	2.13 min	$2 * \Sigma$
FedALA	7+116 min	1.93 min	$2 * \Sigma$

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, Haibing Guan, FedALA: Adaptive Local Aggregation for Personalized Federated Learning, Proceedings of the 37th AAAI Conference on Artificial Intelligence, 2022.

联邦学习异构性---系统异构性

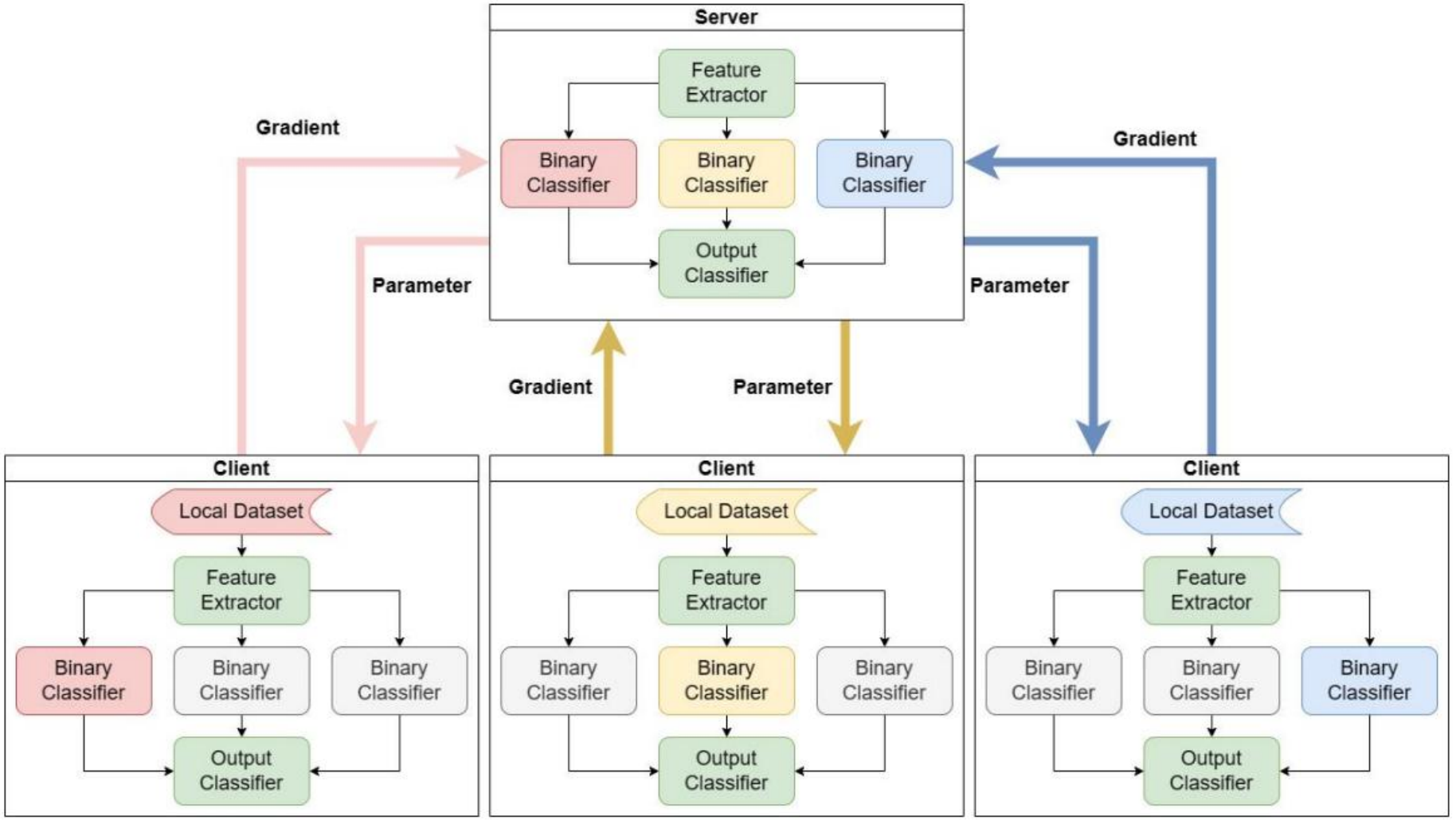
- 系统异构
  - 各客户端计算能力不同，通信开销大
- 适应性训练调整
  - 采用不同模型不同训练参数
  - 采用异步或者回合制聚合





联邦学习异构性---系统异构性

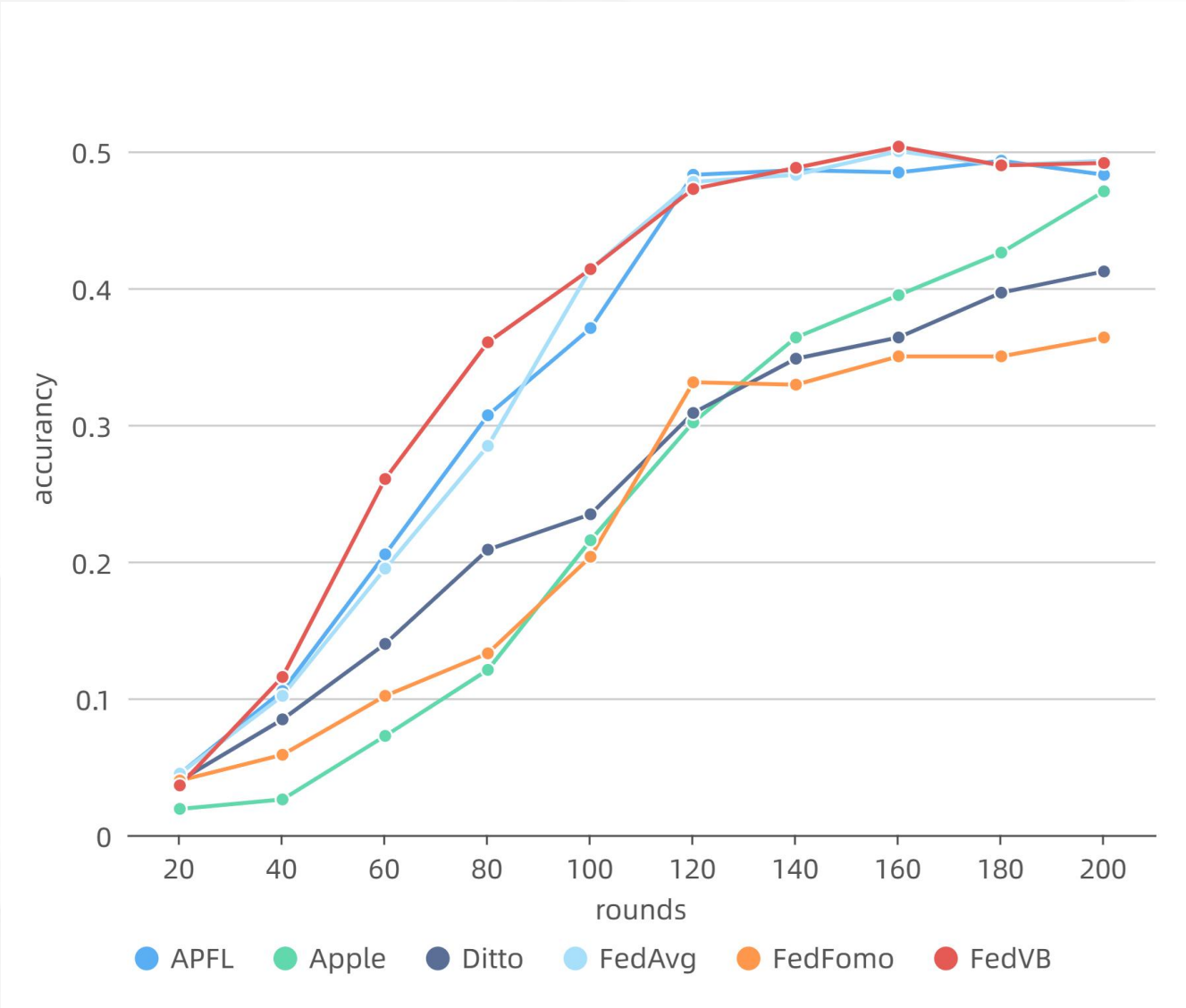
- 将多分类模型分解成多个二分类模型
- 训练时只对**本地标签对应的分支**进行训练
- 传输时只传输**经过训练的模型分支**
- 聚合时只聚合**传输上来的对应分支**



联邦学习异构性---系统异构性

- FedVB相比其他baseline收敛更快，准确率更高
- 多分支模型相比传统模型能够在准确率上有所提升

Algorithm	Simple classifier	Multi-branch Network
FedAvg	0.3983	0.5000
APFL	0.4276	0.5000
Apple	0.3983	0.4707
Ditto	0.3137	0.4259
FedBN	0.3983	0.5017
FedMTL	0.3224	0.5017
FedProx	0.4000	0.5017
pFedMe	0.0259	0.0379
FedFomo	0.8897	0.8500
LOCAL	0.8638	0.8621
PerAvg	0.6000	0.7483



Yunhui Wang, Weichu Zheng, Zifei Liu, Jinyan Wang, Hongjian Shi, Minyu Gu, Ruhui Ma, and Haibing Guan, A Federated Network Intrusion Detection System with Multi-branch Network and Vertical blocking Aggregation, 2023.

Q&A

饮 水 思 源　爱 国 荣 校